JOHANNES KEPLER
UNIVERSITY LINZ

Submitted by
Carlos Eduardo Cancino Chacón

Submitted at
Institute of Computational Perception

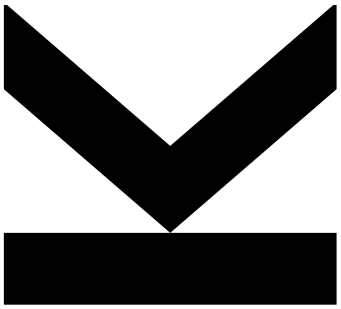
Supervisor and
First Examiner
Gerhard Widmer

Second Examiner
Roger B. Dannenberg

Co-Supervisor
Maarten Grachten

November 2018

Computational Modeling of Expressive Music Performance with Linear and Non-linear Basis Function Models



Doctoral Thesis

to obtain the academic degree of

Doktor der technischen Wissenschaften

in the Doctoral Program

Technische Wissenschaften

JOHANNES KEPLER
UNIVERSITY LINZ
Altenbergerstraße 69
4040 Linz, Österreich
www.jku.at
DVR 0093696

Eidesstattliche Erklärung

Ich erkläre an Eides statt, dass ich die vorliegende Dissertation selbstständig und ohne fremde Hilfe verfasst, andere als die angegebenen Quellen und Hilfsmittel nicht benutzt bzw. die wörtlich oder sinngemäßentnommenen Stellen als solche kenntlich gemacht habe. Die vorliegende Dissertation ist mit dem elektronisch übermittelten Textdokument identisch.

Linz, am

(Unterschrift)

Abstract

This thesis gives a comprehensive overview of the Basis Function Models (BMs), a family of computational models of expressive music performance. These models have been developed over the past years, and have been steadily growing in complexity. The motivation for this work is to model the complex relationship between properties and structure of a given composition, and musically plausible ways of playing the piece expressively and in this way also to learn more about this complex art. The focus is on Western classical music – mostly on the piano, but also with recent extensions towards complex orchestral pieces.

The basic idea in the BM framework is that structural properties of a musical piece (given as a score), which are believed to be relevant for performance decisions, can be modeled in a simple and uniform way, via so-called *basis functions*: numeric features that capture specific aspects of a musical note and its surroundings. A predictive model of performance can then predict appropriate patterns for expressive performance dimensions such as tempo, timing, dynamics, and articulation from these basis functions. A central methodological principle in this work is to take a data-driven approach: the model is not constructed manually, based on musical knowledge or hypotheses, but is learned from a large collection of real human performances, via state-of-the-art linear and non-linear machine learning algorithms. In this way, it is the empirical data that dictates what the model will look like, and an analysis of the learned models can provide interesting insights into the complex relation between score and performance.

A series of BMs of growing complexity will be described. The models are evaluated on corpora of classical piano and symphonic music recordings, in terms of their ability to accurately predict a performer's actual choices. In addition, some qualitative insights gained from an analysis of the models will be presented. Furthermore, recent developments towards integrating the basis function model into a reactive, real-time accompaniment system will be described. This work concludes with a critical and comprehensive survey of the current state-of-the-art in computational models of expressive performance.

Kurzfassung

Diese Dissertation präsentiert einen umfangreichen Überblick über die sogenannten *Basis Function Models of Expressive Music Performance (BM)*, eine Familie von Computermodellen zur Beschreibung und Vorhersage von verschiedenen Dimensionen ausdrucksvoller musikalischer Interpretationen. Ausdrucksvolle oder expressive Interpretation bezeichnet hier das Gestalten von Parameterdimensionen wie Tempo, Timing, Dynamik und Artikulation durch InterpretInnen (z.B. PianistInnen) über die im Notentext vorgegebenen Vorschriften hinaus, um einem Musikstück einen bestimmten musikalischen und affektiven Ausdruck zu verleihen. Ziel dieser *Basis Function Models* ist es, die komplexen Beziehungen zwischen strukturellen Eigenschaften einer Komposition und musikalisch plausiblen Interpretationen des Stücks zu beschreiben.

Die Grundidee der Modelle besteht darin, die für die Gestaltung einer expressiven Interpretation relevanten strukturellen Eigenschaften eines Notentextes in Form sogenannter Basisfunktionen zu repräsentieren. Diese Basisfunktionen sind numerische Repräsentationen (*Feature Functions*), die jeweils spezifische Aspekte von Noten und deren Kontext codieren. Auf der Basis solcher Features soll ein Modell dann aussagekräftige bzw. musikalisch 'adäquate' Muster in Dimensionen wie Dynamik, Tempo und Artikulation vorhersagen.

Die Parameter der Modelle werden nicht mittels händisch erzeugter musiktheoretisch-basierter Regeln definiert, sondern aus einer großen Sammlung echter Interpretationen gelernt. Dies geschieht durch maschinelles Lernen (*Machine Learning*), welches sich moderner linearer und nicht-linearer mathematischer Modelle und dazu passender Optimierungsalgorithmen bedient. Auf diese Weise soll sichergestellt werden, dass die erlernten Modelle eine gute Approximation der echten Interpretationen sind; dadurch sollen interessante Einblicke in die komplexen Beziehungen zwischen Musikstücken und deren Interpretationen ermöglicht werden.

Die vorliegende Dissertation beschreibt eine Reihe von Basisfunktions-Modellen steigender Komplexität – von einfachen linearen Modellen bis hin zu hochgradig nichtlinearen und kontextsensitiven Modellen, zB. in Form von rekurrenten neuronalen Netzwerken mit vielen Ebenen (*Deep Neural Networks*). Die Fähigkeit der Modelle, die interpretativen Entscheidungen eines Musikers präzise vorherzusagen, wird anhand echter Aufnahmen von professionellen Interpreten evaluiert. Der Fokus liegt dabei auf westlicher klassischer Musik – vor allem Klaviermusik –, es wird aber auch eine Erweiterung für Orchestermusik eingeführt. Des weiteren wird ein prototypisches System beschrieben, welches in der Lage ist, einen Solisten in Echtzeit zu begleiten und dabei sowohl seine eigenen Interpretationsstrategien einzubringen als auch sich an bestimmte Spielmuster des Solisten anzupassen. Abschließend wird ein systematischer und kritischer Literaturüberblick über die aktuelle Forschung zur maschinellen Musikinterpretation gegeben, wie sie sich zum heutigen Zeitpunkt darstellt.

Acknowledgments

My doctoral studies would not have been possible without the direct support of many people. First and foremost, I would like to thank Gerhard Widmer, my supervisor, for his support and guidance throughout the course of my doctoral degree. On Friday October 11 2013 at 14:31, I wrote him an email expressing my interest in a PhD position. At 14:45, merely 14 minutes afterwards, he wrote me back and asked about my background and research interests. He then asked if I could come to Vienna on Friday October 18 for an interview, and that very day, I was offered a position at OFAI.

I would like to thank Roger Dannenberg, my second examiner, for his time and effort to review this work. I also thank Werner Goebl for agreeing to be the fourth reviewer of the defense, as well as for his expertise on performance research, which resulted in our collaborations on the development of the ACCompanion, an expressive accompaniment system, as well as the large literature review on models of expressive performance, both of which are documented in this work.

I would like to extend a very special thanks to Maarten Grachten. In many ways, it feels awkward to call the contributions of this work as mine without acknowledging his close supervision and collaboration. Additionally, he introduced me to Emacs and Python, and the joy of open source software, which I heavily used while conducting the experiments documented in this work. I am almost certain that, given enough time, Emacs and a terminal emulator, Maarten could conquer the world.

There are not enough words to thank Laura Bishop. I do not think I would have completed this thesis without her constant support and patience in the last months. Thank you for being my Doctor. I love you heaps and heaps.

I would like to thank my colleagues at OFAI in Vienna and CP/JKU in Linz, who collaborated with me in publications during the course of my doctoral studies. In particular, I would like to thank Stefan Lattner, for fruitful discussions on machine learning models and representations of music information, which resulted in our first publications as doctoral students, as well as for being a very nice officemate; Thassilo Gadermaier, who contributed extensively to the extension of the Basis Function Models for Orchestral Music and building the RCO/Symphonic dataset; and David Sears, for his collaboration and expertise in our study on the role of cognitively plausible information theoretic features and their role in expressive performance. I would also like to thank the rest of the ACCompanion Development Team: Amaury Durand, Martin Bonev and Andreas Arzt. A special thanks goes to Jan Schlüter for keeping me up-to-date with recent developments in machine learning, and being an immense source of expertise in deep learning. I would like to thank my colleagues from the Lrn2Cre8 project, in particular Kat Agres, Gissel Velarde, David Meredith and Geraint Wiggins for very interesting discussions about music, artificial intelligence and computational creativity. A special thanks to Anders Friberg, for the interesting discussions about computational models of expressive performance.

In addition to the above mentioned people, I would also like thank my colleagues of the Intelligent Music Processing and Machine Learning Group at OFAI: Arthur Flexer, Sebastian Böck, Silvan Peter, Martin Gasser, Roman Feldbauer and Thomas Grill, for the many conversations, ideas, nice lunches, and helping me to sort some of the idiosyncrasies of Austrian bureaucracy. Also, I would like to thank my colleagues from the CP in Linz: Andreu Vall, Mathias Dorfer, Filip Korzeniowski, Rainer Kelz and Richard Vogl, for the interesting discussions on MIR and deep learning, as well as good times during conferences. Last but not least, a big thanks to Karin

Pils-Vorsteher, Gudrun Schuchmann, Constantin Marsch and Inge Hauer, for keeping OFAI running smoothly (and making research much easier).

I would also like to thank my friends and family for their continued support. In particular, I want to thank my parents and my siblings for always being there for me and giving me their unconditional love and support. I want to thank Mario Watanabe for the very interesting conversations about life, the universe and everything. I also want to thank my Austrian-Canadian friends Micheline Welte, Milena Bolaños and Alexander von Franqué for all the nice dinners, interesting discussions about pop culture and a lot of fun, including the weekly (and infinitely enjoyable) baking nights, the yearly quests to get the most comics during Free Comic Book Day and Canadian Thanksgiving celebrations. A special thanks goes to my Mexican friends Rafael Cruz and José Antonio Maqueda, for being my best friends for more than half of our lives ¡Syninoforcimno!. I would also like to thank Casilda Rivera and Marcos Chacón for being a family to me in these lands so far away from Mexico.

I would like give an acknowledgment to Pi, who I believe to be the only Australian cat in Austria, who slept on top of my laptop for a significant part of the writing of this thesis (and thus, forced me to get an external keyboard). Also, to Zyanya and Fantine.

The research for this work was funded by the European Union Seventh Framework Programme through the Lrn2Cre8 project (FET grant agreement No. 610859) and the European Research Council (ERC) under the European Union's Horizon 2020 Framework Programme (ERG Grant Agreement No. 670035, project Con Espressione).

This thesis is respectfully dedicated to the memory of Mr. Elias Howe, who in 1846, invented the sewing machine.

Contents

List of Figures xiv

List of Tables xix

List of Algorithms xxi

1 Introduction 1
- 1.1 Motivation . . . 1
  - 1.1.1 A Brief Historical Overview of the Study of Performance . . . 1
  - 1.1.2 Computational Methods of Expressive Performance . . . 2
  - 1.1.3 This Work . . . 3
- 1.2 Contributions . . . 4
- 1.3 Publications . . . 5
- 1.4 Thesis Outline . . . 7

2 Computational Models of Expressive Performance 10
- 2.1 Introduction . . . 10
- 2.2 Strategies for Modeling Musical Expression . . . 12
  - 2.2.1 Rule-Based vs. Data-Driven Modeling . . . 12
  - 2.2.2 Analysis vs. Synthesis . . . 13
- 2.3 Components of a Computational Model of Expressive Performance for Notated Music . . . 14
  - 2.3.1 Score Features . . . 15
  - 2.3.2 Expressive Parameters . . . 16
  - 2.3.3 Computational Models . . . 16
- 2.4 Datasets Used in this Thesis . . . 17
  - 2.4.1 Batik/Mozart . . . 18
  - 2.4.2 Magaloff/Chopin . . . 18
  - 2.4.3 Zeilinger/Beethoven . . . 19
  - 2.4.4 RCO/Symphonic . . . 19

3 Basis Function Models I: Linear Models 21
- 3.1 Introduction . . . 21
- 3.2 The Basis Function Models: A Formal Description . . . 21
  - 3.2.1 Expressive Parameters . . . 22
  - 3.2.2 Score Features: Score Basis Functions . . . 26
  - 3.2.3 Computational Model . . . 28
  - 3.2.4 A Note on Evaluation Strategies for Computational Models of Musical Expression . . . 31
- 3.3 Linear Basis Models . . . 31
  - 3.3.1 Expressive Parameters: Performance Codec v. 1.0 . . . 32
  - 3.3.2 Basis Functions . . . 36
  - 3.3.3 Model Description . . . 39

3.4 Bayesian Linear Basis Models . . . 40
3.4.1 Basis Functions . . . 41
3.4.2 Model . . . 41
3.5 Evaluation . . . 43
3.5.1 Goodness of fit: Expressive Dynamics . . . 44
3.5.2 Predictive Accuracy: Expressive Dynamics . . . 45
3.5.3 Predictive Accuracy: All Expressive Parameters . . . 45
3.6 Discussion . . . 46
3.7 Conclusions . . . 48

4 Basis Function Models II: Non-linear Models 50
4.1 Introduction . . . 50
4.2 Non-Linear Basis Models . . . 51
4.2.1 Basis Functions . . . 51
4.2.2 Model Description . . . 52
4.2.3 Evaluation: Expressive Dynamics . . . 55
4.2.4 Discussion . . . 59
4.3 Recurrent Non-linear Basis Models . . . 60
4.3.1 Motivation for Sequential Models . . . 60
4.3.2 Expressive Parameters for Sequential Models: Performance Codec v. 2.0 . . . 61
4.3.3 Adapting Basis Functions for Onset-wise and Note-wise Parameters . . . 65
4.3.4 Model Description . . . 67
4.3.5 Experiments: Expressive Tempo . . . 71
4.3.6 Results and Discussion . . . 72
4.4 Conclusions . . . 73

5 From Solo Piano to Orchestral Ensembles 76
5.1 Introduction . . . 76
5.2 Basis Functions for Orchestral Music . . . 77
5.2.1 Measured vs. Computed Expressive Parameters . . . 77
5.2.2 Indexing Basis Functions . . . 78
5.2.3 Merging and Fusion of Basis Functions within Instrument Classes . . . 78
5.2.4 Aggregation of Basis Functions of Instrument Classes in a Piece . . . 80
5.3 Experiments: Expressive Dynamics . . . 80
5.3.1 Expressive Parameter . . . 81
5.3.2 Model Architectures and Training . . . 81
5.3.3 Predictive Accuracy . . . 82
5.4 The BM as an Analytical Tool for Dynamics in Symphonic Music . . . 82
5.5 Conclusions . . . 85

6 The Role of Cognitively Motivated Features for Expressive Performance 87
6.1 Introduction . . . 87
6.2 Related Work . . . 88
6.2.1 Musical Expectation . . . 88
6.2.2 Tonal Tension . . . 88
6.3 Modeling Expressive Performances . . . 89
6.3.1 Expressive Parameters . . . 89
6.3.2 Basis Functions . . . 91
6.3.3 Computational Model . . . 95
6.4 Experiments . . . 95

6.5 Results and Discussion . . . 97
6.5.1 Predictive accuracy . . . 97
6.5.2 Sensitivity Analysis . . . 99
6.6 Conclusions . . . 100

7 Implementing the Basis Function Models: The Basis Mixer 102
7.1 Introduction . . . 102
7.2 Overview of the Basis Mixer . . . 103
7.2.1 Rendering a Performance . . . 103
7.2.2 Training the Basis Mixer . . . 105
7.3 Performance Representation . . . 107
7.3.1 Encoding a Performance . . . 108
7.3.2 Decoding a Performance . . . 109
7.4 Score Representation . . . 110
7.4.1 Implementation Details . . . 110
7.5 Predictive Function . . . 111
7.5.1 Functions for Onset-wise Parameters . . . 111
7.5.2 Functions for Note-wise Parameters . . . 111
7.5.3 Implementation Details . . . 111
7.6 Interactive Web Interface . . . 112
7.7 Conclusions . . . 113

8 Using Basis Function Models for Expressive Accompaniment: The ACCompanion 115
8.1 Introduction . . . 115
8.2 Score Following . . . 117
8.2.1 Score Following for Expressive Accompaniment . . . 117
8.2.2 HMM-based Score Follower . . . 118
8.3 Adapting the Basis Function Models for Expressive Accompaniment . . . 120
8.3.1 Precomputing the Performance of the Accompaniment Score . . . 121
8.3.2 Capturing the Local Performance Style of the Soloist . . . 123
8.3.3 Adapting the Performance of the Accompaniment Part to the Soloist . . . 125
8.4 The ACCompanion . . . 127
8.4.1 Implementation Details . . . 129
8.4.2 Methods for Demonstrating and Visualizing the ACCompanion . . . 129
8.4.3 AccompaniX 2017 . . . 132
8.4.4 Videos for ISMIR 2017 . . . 132
8.5 Conclusions . . . 133

9 In Conclusion: A Comprehensive and Critical Review of the Current State of the Field 135
9.1 Introduction . . . 135
9.2 Computational Modeling of Expressive Performance . . . 136
9.2.1 Motivations for Computational Modeling . . . 136
9.2.2 Components of the Performance Process . . . 137
9.2.3 Components of Computational Models . . . 138
9.3 A Synopsis of Current State and Recent Trends . . . 139
9.3.1 Data-driven Methods for Analysis and Generation of Expressive Performances . . . 139
9.3.2 Expressive Interactive Systems: Models for Technology-mediated Performance . . . 144
9.3.3 Use of Cognitively Plausible Features and Models . . . 148

9.3.4 New Datasets . . . 149
9.3.5 Computational Models as Tools for Music Education . . . 152
9.4 A Critical Discussion of Parameter Selection and Model Evaluation . . . 153
9.4.1 Encoding Expressive Dimensions and Parameters . . . 153
9.4.2 Relation to Palmer's Categories . . . 156
9.4.3 Evaluating Computational Performance Models . . . 159
9.5 Conclusions . . . 161

Bibliography **164**

A Basis Functions **185**
A.1 List of Basis Functions . . . 185
A.1.1 Pitch . . . 185
A.1.2 Metrical . . . 186
A.1.3 Notated Performance Directives . . . 187
A.1.4 Music Theoretic Features . . . 190
A.1.5 Expectancy Features . . . 191
A.1.6 Tonal Tension Features . . . 192
A.2 List of Orchestral Basis Functions . . . 193
A.3 Instrument List . . . 204

B Towards a Perceptually Plausibly Description of Expressive Tempo **205**

C Modeling Joint Interactions of Expressive Parameters **208**
C.1 Introduction . . . 208
C.2 Evidence for Interactions from Music Psychology . . . 209
C.3 Empirical Analysis of Recorded Piano Performances . . . 210
C.4 Joint Modeling of Expressive Parameters . . . 211
C.4.1 Joint Modeling of Expressive Parameters Using NBMs . . . 212
C.4.2 Experiments . . . 212
C.4.3 Results and Discussion . . . 213
C.5 Conclusions . . . 215

D Probabilistic Non-linear BMs using GMDNs **218**
D.1 Introduction . . . 218
D.2 Model Description . . . 219
D.3 Experiments . . . 220
D.3.1 Model Architectures and Training . . . 221
D.4 Results and Discussion . . . 221
D.5 Visualizing Interactions Between Expressive Parameters . . . 223
D.6 Conclusions . . . 224

E Constructing and Training Linear Models **226**
E.1 Standard Least Squares . . . 226
E.2 Regularized Least Squares . . . 227
E.3 Derivation of the Weights for Bayesian Linear Regression Using a Gaussian Prior 227
E.4 Estimation of the Prior Parameters using the EM Algorithm . . . 228

F Building and Training Neural Network Models **238**
F.1 RMSProp . . . 238
F.2 Recurrent Layers Used in this Work . . . 239
F.2.1 Vanilla Recurrent Layer . . . 239

F.2.2 LSTM with Multiplicative Integration 239

G Sampling from Generative Probabilistic Models **240**
G.1 Introduction 240
G.2 Linear Basis Models 240
G.3 (Recurrent) Non-linear Basis Models 241

H Mathematical Formulae and Identities **244**
H.1 Probability Theory 244
H.2 Linear Algebra 244
H.3 Miscellaneous 245

Curriculum Vitae of the Author **247**

List of Figures

2.1 Tom Cat (top left) vs. Bugs Bunny (top right) performing Liszt's Hungarian Rhapsody No. 2. The plot in the bottom left compares the local tempo (in bpm) of both performances. The plot in the bottom right compares the performed dynamics (measured as the RMS energy of the signal normalized to lie in the interval $[0, 1]$). The images of Bugs Bunny and Tom Cat are property of Warner Bros. Entertainment Inc. 11

2.2 First 16 bars of the arrangement of Liszt's Rhapsody No. 2 as performed by Bugs Bunny (my transcription based on Edition Peters (Liszt, 1913)). The heatmap shows the tempo curve of Bug's performance (in red in the lower left plot in Figure 2.1). 12

2.3 Framework for computational models of expressive performance. 14

3.1 A basis function model for modeling expressive performances: Aspects of a score $\mathfrak{S}$ are represented numerically by $\mathbf{\Phi}$, using a score representation model $\mathcal{F}$. Aspects of an expressive performance are captured using a performance codec $\mathcal{C}$, consisting of a performance representation model $\mathcal{Y}$ and a performance decoder $\mathcal{Y}^{-1}$. The score representation $\mathbf{\Phi}$ serves as input to a predictive function $F(\cdot, \boldsymbol{\theta})$, which outputs $\mathbf{Y}$, a prediction of a performance representation under representation model $\mathcal{Y}$. Using a performance decoder $\mathcal{Y}^{-1}$, we can generate a matched performance $\mathfrak{P}_{\mathfrak{S}}$. $\mathbf{\Phi}$, $\mathbf{Y}$ and F correspond to the score features, expressive parameters and computational model described in Section 2.3. See text for a detailed explanation of these concepts. 22

3.2 Piano roll of performance matched onsetwise $\mathfrak{P}_{\mathfrak{S}}$(left) and information describing the performance for each note (right). The intensity of the hue of the notes on the piano roll represents their MIDI velocity. Functions $\text{vel}(\cdot)$, $\text{onset}_{perf}(\cdot)$ and $\text{duration}_{perf}(\cdot)$ represent the performed MIDI velocity, onset time and duration (in seconds) of each note in the score $\mathfrak{S}$. 23

3.3 Performance space $\mathcal{P}$ spanned by performance representation model $\mathcal{Y} = \{\mathsf{y}_{\text{vel}}, \mathsf{y}_{\text{bpm}}\}$ for the matched performance shown in Figure 3.2. Each point in the performance space correspond to the representation of the performance of a score note n, i.e. a row in performance representation $\mathbf{T}$. 25

3.4 Elements of Score $\mathfrak{S}$. Score notes are $\boldsymbol{\mathcal{X}} = \{n_1, \ldots, n_{15}\}$ and score onsets $\boldsymbol{\mathcal{O}} = \{o_1, \ldots, o_{11}\}$. 26

3.5 Score space $\mathcal{S}$ spanned by score representation model $\mathcal{F} = \{\varphi_{\text{pitch}}, \varphi_{\boldsymbol{f}}\ \}$ for the score shown in Figure 3.4. Each point in the space correspond to the representation of a score note n, i.e. a row of $\mathbf{\Phi}$. Note that due to the chosen representation model, the points for some of the notes overlap. 28

3.6 Comparison between predicted ($\mathbf{Y}$, in blue) and performed ($\mathbf{T}$, in red) expressive parameters for the performance representation model $\mathcal{Y} = \{\mathsf{y}_{\text{vel}}, \mathsf{y}_{\text{bpm}}\}$ described in Figure 3.3. The predictions were computed using a linear model with the score representation $\mathcal{F} = \{\varphi_{\text{pitch}}, \varphi_{\boldsymbol{f}}\ \}$ described in Figure 3.5. 29

3.7 Excerpt of a matched MIDI performance $\mathfrak{P}_{\mathfrak{S}}$ (as a piano-roll where the x axis describes performance time and the y axis describes the MIDI velocity) and its corresponding score $\mathfrak{S}$ showcasing elements for computing the expressive parameters described in performance codec v. 1.0. 32
3.8 Performance representation $\mathbf{T}$ under representation model $\mathcal{Y}_{1.0}$ for matched performance $\mathfrak{P}_{\mathfrak{S}}$ (with score $\mathfrak{S}$) introduced in the running example in Section 3.2 (see Figure 3.2). 35
3.9 Left: Schematic view of basis functions representing score information for score $\mathfrak{S}$. Right: $\mathbf{\Phi}$, the score representation of $\mathfrak{S}$ under score representation model $\mathcal{F} = \{\varphi_{\boldsymbol{p}}, \varphi_{\boldsymbol{f}}, \varphi_{\boldsymbol{cresc}}, \varphi_{\text{accent}}\}$. 36
3.10 Motivation for the polynomial pitch model: polynomial fitting of the Magaloff/Chopin dataset. 38
3.11 Comparison of the predictions of the expressive parameters made by the BLBM (in blue) Magaloff's performance (in red) for Prelude in B major Op. 28, No. 11 by F. Chopin. 47

4.1 Schematic representation of the prediction of an expressive parameter using an NBM consisting of an FFNN with two layers. From bottom to top, the circles represent the input layer, two successive hidden layers and the output layer, respectively; $\boldsymbol{\varphi}(n_i)$ represents the input basis functions for score note n_i, $\boldsymbol{h}_i^{(1)}$ and $\boldsymbol{h}_i^{(2)}$ are the activations of the first and second hidden layers for score note n_i, respectively; and y_{n_i} is the output of the network for score note n_i. The arrows connecting circles represent the flow of information in the FFNN. Note that there are no connections between neighboring notes – the performance of each note is predicted independently of the others, by the same trained FFNN. See Section 4.2.2 for a more detailed explanation. 53
4.2 Example of the effect of the interaction of *crescendo* after a *diminuendo* for both LBM and NBM. 60
4.3 Excerpt of a matched MIDI performance $\mathfrak{P}_{\mathfrak{S}}$ (as a piano-roll where the x axis describes time and the y axis describes the MIDI velocity) and its corresponding score $\mathfrak{S}$ showcasing elements for computing the expressive parameters described in performance codec v. 2.0. 62
4.4 Performance representation $\mathcal{Y}_{2.0}(\mathfrak{S} \mid \mathfrak{P}_{\mathfrak{S}}) = \{\mathbf{T}_{\text{note-wise}}, \mathbf{T}_{\text{onset-wise}}\}$ for matched performance $\mathfrak{P}_{\mathfrak{S}}$ (with score $\mathfrak{S}$) for the running example introduced in Section 3.2 (see Figure 3.2). Colored rectangles highlighting regions of $\mathbf{T}_{\text{note-wise}}$ and $\mathbf{T}_{\text{onset-wise}}$ correspond to the score position highlighted with the same color. . . . 65
4.5 Note-wise ($\mathbf{\Phi}_{\text{note-wise}}$) and onset-wise ($\mathbf{\Phi}_{\text{onset-wise}}$) components of the score representation of $\mathfrak{S}$ under score representation model $\mathcal{F} = \{\varphi_{\boldsymbol{p}}, \varphi_{\boldsymbol{f}}, \varphi_{\boldsymbol{cresc}}, \varphi_{\text{accent}}\}$. Colored rectangles highlighting regions of $\mathbf{\Phi}_{\text{note-wise}}$ and $\mathbf{\Phi}_{\text{onset-wise}}$ correspond to the score position highlighted with the same color. 68
4.6 Schematic representation of the prediction of an expressive parameter using an RNBM consisting of an RNN with a single bidirectional layer. From bottom to top, the circles represent the input layer, the forward and backward components of the bidirectional layer, and the output layer, respectively ($\boldsymbol{\varphi}(o_i)$ represents the input basis functions for score onset o_i, $\boldsymbol{h}_{i;\text{fw}}$ and $\boldsymbol{h}_{i;\text{bw}}$ are the activations of the forward and backward components of the recurrent hidden layer for score onset o_i, respectively; and y_{o_i} is the output of the network for score note o_i). From left to right, advancing time steps are shown. The arrows connecting circles represent the flow of information in the RNN. See Section 4.3.4 for a more detailed explanation. 69

4.7 Sensitivity of log BPR predictions to Fermata, Duration and Accent basis functions, for the ten models obtained from 10-fold cross-validation. Positive values on the vertical axis denote a lengthening of BPR (tempo decrease), negative values a shortening (tempo increase). The curves show the effect of the activation of the basis function at time step $j = 0$ on log BPR values (see Equation (4.36)) in the past (negative time steps), and future (positive time steps). 74

5.1 Motivation for *merging* and *fusion*. Different pieces' scores will in general have different number of instances of the same instrument class (e.g. 4 vs. 8 Horns). Carefully note the different possibilities to distribute voices across staves: in the right hand score excerpt, the lower staff of the trumpet carries two instances, whereas the upper staff only one. Similar for the trombone. The instances of one instrument class (e.g. 8 horns, right) will be *merged* to one representative to be matched to the representative from the other score (subsuming 4 horns, left). . . 79

5.2 Illustration of *merging* and *fusion* of score information of two different parts belonging to the same instrument class "Oboe". The first matrix (top left) shows three example basis functions, φ_{pitch}, $\varphi_{\text{duration}}$, and $\varphi_{\boldsymbol{f}}$, for the first few notes of each of the two score parts. Note the interleaved, consecutive layout of simultaneously occurring notes, as indicated by the first column of each matrix, giving the notes' onset times. The matrix top right illustrates *merging*, where the data of both score parts were combined to give one column per basis function. Finally, the third matrix is the result of *fusion* operations, applied per basis function to each set of values occurring at the same time point. The fusion operation used takes the average of the respective values. See text for further explanation. . . . 80

5.3 Aggregating the per instrument class matrices into a per piece matrix. Note how the Clarinet has different dynamics prescribed (so that the ***forte*** basis function is all zeros). The Bassoon is obviously silent in this part of the piece. 81

5.4 Top: Measured and fitted loudness curves for an excerpt of the performance of Beethoven's 6th Symphony, 3rd Movement (bars 87 to 92) by Harnoncourt (orange) and Solti (purple). Bottom: sensitivity-difference graph for Harnoncourt and Solti. Orange tones indicate that a basis function has a stronger (positive) contribution to loudness in Harnoncourt than in Solti; purple tones indicate the opposite. 85

6.1 Representation of non-enharmonically equivalent pitch classes in the spiral array. 93

6.2 Normalized mutual information between features and expressive parameters. A larger mutual information means that the variables are more related to each other. 97

6.3 Sensitivity plots for BPR (top left), dBPR (top right), VEL (bottom left) and dVEL (bottom right). Each row in the plot corresponds to an input feature and each column to the contribution of its value at that time step to the output of the model at τ (the center of each plot). Red and blue indicate a positive and negative contribution, respectively. 101

7.1 Schematic representation of the relationship between components of the Basis Mixer for rendering the performance of a piece given a score $\mathfrak{S}$. Pink rectangles with rounded corners and thick black borders represent instances of the main components of the system: `BasisGenerator`, `RegressionModel`, and `PerformanceCodec`. The Yellow rectangle with a thin black border represents a method for post-processing the predicted performance $\mathbf{Y}$: rescale and recenter and use the specified average BP. Black arrows denote the direction of the flow of information. Blue arrows denote sharing of configuration information passed to the constructor of the instances of the main components. The background colored rectangles without a border encapsulate the components of the Basis Mixer that correspond to the main components of the BM framework: the score $\mathfrak{S}$ (in orange), the performance $\mathfrak{P}_{\mathfrak{S}}$ (in light yellow), the score representation model $\mathcal{F}$ (in green), the performance codec $\mathcal{C}$ (in red), and the predictive function F (in blue). 104

7.2 Schematic representation of the relationship between components of the Basis Mixer for training the predictive function given a test set $\boldsymbol{\mathcal{T}}$. Pink rectangles with rounded corners and thick black borders represent instances of the main components of the system: `BasisGenerator`, `RegressionModel`, and `PerformanceCodec`. Black arrows denote the direction of the flow of information. Blue arrows denote flow of configuration information passed to the constructor of the instances of the main components. The background colored rectangles without a border encapsulate the components of the Basis Mixer that correspond to the main components of the BM framework: the score $\mathfrak{S}$ (in orange), the performance $\mathfrak{P}_{\mathfrak{S}}$ (in light yellow), the score representation model $\mathcal{F}$ (in green), the performance codec $\mathcal{C}$ (in red), and the predictive function F (in light blue). 106

7.3 Excerpt of the first movement of Sonata KV. 332 by W. A. Mozart performed by Roland Batik in Matchfile format. Each line represents a score note (highlighted in pink in the first line) and a performance note (highlighted in blue for the first line). The score note information includes attributes such as the index of the (see `n2`, which corresponds to n_2 in the notation used in this thesis, highlighted in gold), its notated pitch and octave (F natural 4 , highlighted in brown and gray, respectively for n_2). The performed note information includes the number of the note in the MIDI file (2 highlighted in red for n_2), the performed onset time (in MIDI ticks, highlighted in light green for n_2); key-off time and the sound-off time estimated using pedal information (highlighted in purple and yellow for n_2, respectively); and the performed MIDI velocity of the note (highlighted in light blue). Performed notes that do not represent notes in the score are denoted as insertions (highlighted in orange). Omitted notes are denoted as deletions (see the omission of n_{933} highlighted in green). 109

7.4 Prototype UI for the Web App. 113

8.1 Schematic representation of the KHMM score follower. O_i (in pink) is a random variable representing the onset at observation i. BP_i (in blue) is a random variable representing the performed BP at observation i. IOI_i (in orange) is a random variable representing the observed IOI at observation i. P_i (in light green) is a random variable representing the observed MIDI pitch at observation i. Square nodes represent discrete variables and circle nodes represent continuous variables. The gray boxes denote the latent and observed variables. See text for explanation. 121

8.2 Schematic representation of the modules of the ACCompanion. Black arrows represent the flow of information for computing the accompaniment part. The thick orange arrow represents the implicit effect of the system output on the (live) soloist. 127

8.3 Screenshot of the performance visualizer of the ACCompanion. 131
8.4 Thumbnails of the videos demonstrating the capabilities of the ACCompanion. On the left: Werner Goebl performing the right hand part of second movement of the Sonata No. 16 KV. 545 by W. A. Mozart. On the right: Gerhard Widmer performs the solo part of The Wild Geese. The piano rolls represent the visualization described in Section 8.4. These videos were recorded at the Fanny Hensel-Mendelssohn Hall at the University of Music and Performing Arts Vienna. The piano is a Bösendorfer CEUS 280VC. 133
8.5 Setup for recording the demonstration videos at of the ACCompanion. In the picture (left) Gerhard Widmer, Werner Goebl (at the piano) and myself (in front of the computer). 134

9.1 Computational Modeling Scenario. 138

A.1 Excerpt of the Sonata No. 18 Op. 31 No. 3 by L. van Beethoven. The colored rectangles/circles highlight notated performance directives. Purple rectangles highlight instances of ***p***; yellow rectangles highlight instances of legato slurs; green rectangles highlight instances of ***crescendo***, red rectangles highlight instances of ***ritardando*** (whose effect ends in the *a tempo* markings, highlighted in gray); little blue circles highlight staccato markings; pink rectangles highlight fermatas; light blue rectangles highlight instances of ***sf*** markings; an orange rectangle highlights an instance of **Allegro**; a violet rectangle highlights an instance of a grace note. The excerpt from the score comes from the Breitkopf & Härtel edition (van Beethoven, 1862). 188

B.1 Excerpt of a matched MIDI performance $\mathfrak{P}_{\mathfrak{S}}$ (as a piano-roll where the x axis denotes time and the y axis denotes MIDI velocity) and its corresponding score $\mathfrak{S}$ showcasing elements for computing the cognitively motivated expressive parameters relating to tempo, timing and articulation. 206
B.2 Comparison of $\log \mathrm{BPR}$ computed using performance codecs v. 1.0 and 1.5 for the first bars of Chopin's Nocturne Op. 9 No. 1 in B♭ minor performed by Nikita Magaloff. The red vertical dotted lines denote barlines. The excerpt from the score comes from Schirmer's Edition (Chopin, 1915a). 207

C.1 Spearman's ρ between expressive targets, measured from the Magaloff/Chopin corpus (top), and the Zeilinger/Beethoven corpus (bottom) 211

D.1 Visualization of the probability density function for a GMDN modeling note-wise parameters for an excerpt of Chopin's Prelude Op. 28 No. 7 in A major. Each color box denotes the distribution for a single note (indicated with the same color in the score). The excerpt from the score is taken from Schirmer's Edition (Chopin, 1915b). 224
D.2 Visualization of the probability density function for a GMDN modeling onset-wise parameters for the last two bars of Chopin's Prelude Op. 28 No. 7 in A major. Each color box denotes the distribution for a single onset (indicated with the same color in the score). Thin grey horizontal and vertical gridlines were added to the plots of the distributions to better compare the location of the mode of the distribution. The excerpt from the score is taken from Schirmer's Edition (Chopin, 1915b). 225

H.1 Comparison of the activation functions used in the neural networks described in the main text. 246

List of Tables

2.1 Musical material contained in the Batik/Mozart corpus 17
2.2 Musical material contained in the Zeilinger/Beethoven corpus 17
2.3 Musical material contained in the RCO/Symphonic corpus 18
2.4 Musical material contained in the Magaloff/Chopin corpus 20

3.1 Goodness-of-fit of the model over performances of four Chopin piano pieces. See Section 3.5 for abbreviations. 45
3.2 Predictive accuracy in a leave-one-out scenario over performances of 151 Chopin piano pieces. See Section 3.5 for abbreviations. 46
3.3 Predictive accuracy of the Bayesian linear regression using feature set DYN+PIT for the expressive parameters defined in performance representation model v. 1.0, described in Section 3.3.1. 46

4.1 Predictive accuracy for expressive dynamics in terms of the coefficient of determination (R^2) and Pearson correlation coefficient (r), averaged across pieces on each corpora. 56
4.2 Basis functions with the largest sensitivity coefficients for the LBM models. Averages are reported over the 5 folds of the cross-validation. Dynamics markings are in bold italic. Basis functions that are non-zero for less than 5% of the instances have been grayed out. 58
4.3 Basis functions with the largest sensitivity coefficients for the NBM models; Averages are reported over the 5 folds of the cross-validation. Dynamics markings are in bold italic. Basis functions that are non-zero for less than 5% of the instances have been grayed out. 59
4.4 Predictive results for IOI, averaged over a 10-fold cross-validation on the Magaloff/Chopin corpus. Larger r and R^2 means better performance. 72

5.1 Predictive accuracy in a leave-one-out scenario for the different models. For both R^2 and r, larger is better. Best value per piece and measure emphasized in bold. 83

6.1 Predictive results for expressive tempo and dynamics, averaged over all pieces on the Batik/Mozart corpus. Larger R^2 and r means more accurate predictions. . . 96
6.2 Proportion of variance explained (R^2) for expressive tempo using different feature sets, averaged over all pieces on the Batik/Mozart corpus. Larger R^2 values reflect more accurate predictions. For each combination of target and feature set, the results are listed for that feature set as is (left), and including expectancy features **E**, tonal tension features **T** or both **E** + **T** (right). For clarity, improvements of R^2 as a result of adding **E**, **T** or both are marked in green, whereas deteriorations are marked in red. Bold numbers mark a statistically significant difference ($p < 0.01$). The effect size (Cohen's d) is reported in parenthesis for those cases with statistically significant differences. 98

6.3 Proportion of variance explained (R^2) for expressive dynamics using different feature sets, averaged over all pieces on the Batik/Mozart corpus. Larger R^2 values reflect more accurate predictions. For each combination of target and feature set, the results are listed for that feature set as is (left), and including expectancy features **E**, tonal tension features **T** or both **E** + **T** (right). For clarity, improvements of R^2 as a result of adding **E**, **T** or both are marked in green, whereas deteriorations are marked in red. Bold numbers mark a statistically significant difference ($p < 0.01$). The effect size (Cohen's d) is reported in parenthesis for those cases with statistically significant differences. 99

9.1 Probabilistic Models for Performance Generation 143
9.2 Neural Network Based Models 145
9.3 Feedback and conductor systems 147
9.4 Datasets of expressive performances. 150

C.1 Predictive results for MIDI velocity for each architecture, averaged over a 5-fold cross-validation on the Magaloff/Chopin and Zeilinger/Beethoven piano performance corpora. A larger R^2 and r means better performance. The numbers in bold represent the best architecture (in terms of R^2) for each condition (Independent or Joint). See text for description of the neural architectures. 214
C.2 Predictive results for log BPR for each architecture, averaged over a 5-fold cross-validation on the Magaloff/Chopin and Zeilinger/Beethoven piano performance corpora. A larger R^2 and r means better performance. The numbers in bold represent the best architecture (in terms of R^2) for each condition (Independent or Joint). See text for description of the neural architectures. 215
C.3 Predictive results for timing for each architecture, averaged over a 5-fold cross-validation on the Magaloff/Chopin and Zeilinger/Beethoven piano performance corpora. A larger R^2 and r means better performance. The numbers in bold represent the best architecture (in terms of R^2) for each condition (Independent or Joint). See text for description of the neural architectures. 216
C.4 Predictive results for log articulation for each architecture, averaged over a 5-fold cross-validation on the Magaloff/Chopin and Zeilinger/Beethoven piano performance corpora. A larger R^2 and r means better performance. The numbers in bold represent the best architecture (in terms of R^2) for each condition (Independent or Joint). See text for description of the neural architectures. 217

D.1 Predictive accuracy for MIDI velocity for GMDFFNNs with different number of components, averaged over a 5-fold cross-validation on the Magaloff/Chopin (top) and Zeilinger/Beethoven (bottom). A larger R^2 and r means better performance. 222

List of Algorithms

3.1 Performance Decoder v. 1.0 37

4.1 Performance Decoder v. 2.0 66

8.1 `accompaniment_step` 128
8.2 ACCompanion v. 0.1 130

E.1 EM Algorithm for estimating $\mathbf{m}_0$, $\mathbf{S}_0$ and β 237

G.1 Approximation of the predictive distribution using importance sampling. 242

1 Introduction

1.1 Motivation

This thesis focuses on the study of expressive music performance through (data-driven) computational methods.

Expressive performance of music constitutes a fundamental part of the enjoyment of several kinds of music, yet musical expression is a complex phenomenon that is not fully understood. Performances of written music by humans are hardly ever exact acoustical renderings of the notes in the score, as a computer would produce. Nor are they expected to be: a natural human performance involves an interpretation of the music, in terms of structure, but also in terms of affective content (Clarke, 1988; Palmer, 1996; Gabrielsson and Lindström, 2010), which is conveyed to the listener by local variations in dimensions such as tempo, loudness, and, depending on the expressive possibilities of the instrument, the timing, articulation, and timbre of individual notes.

Becoming an expert musician takes many years of training and practice, usually in a master-apprentice scenario, and rather than adhering to explicit rules, achieved performance skills are to a large degree the effect of implicit, procedural knowledge (McPherson and Welch, 2011). This is not to say that regularities cannot be found in the way musicians perform music. Decades of empirical research have identified a number of factors that jointly determine the way a musical piece is rendered (Palmer, 1997; Gabrielsson, 2003).

At this point, it is important to note that music performance is a sociocultural construct that has different meanings for different cultures. For example, in Western cultures, a music performer is assumed to have some level of expertise (i.e. not everybody can participate) and audiences often have a more passive role of listening to the performance. On the other hand, in other cultures, like the Venda traditional music from South Africa, the performance of music is a social activity in which everybody participates and the experience of the performance is not only the *acoustic* rendition of the music, but includes the social experience as well (Hill, 2012).

While in many traditions music can be considered an oral-aural activity, i.e. performing traditions are passed on orally (Fabian, 2017), the focus is this thesis is on notated Western art music of the common practice period[1] – mostly on the piano, but also with recent extensions towards orchestral music.

1.1.1 A Brief Historical Overview of the Study of Performance

Music performance has been around since the early days of mankind, with archaeological evidence suggesting that musical instruments were made as far back as 35 000 years ago (Conard, 2009). During most of the history of music in the modern era[2], both musicians and music

[1] The common practice period is usually associated with the second half of the 17th century to the early 20th century, including the late Baroque, Classical, Romantic and Impressionist periods.

[2] According to the traditional linear historiographical approach, the modern era corresponds to the period starting after the middle ages, from the early 16th century onwards (Vovelle, 1986).

scholars have theorized on the way music is (or should be) performed, most importantly for pedagogical reasons (Robertson and Stevens, 1966). Important efforts in this regard include F. Couperin's *The Art of Playing Harpsichord* (published in 1716), L. Mozart's *Treatise on Fundamentals of Violin Playing* (published in 1756) and C. P. E. Bach's *Essay on True Art of Playing Keyboard Instruments* (published in 1787). These treatises focus not only on developing the technical proficiency of the students, but also attempt to describe of how to perform the music expressively according to the conventions of their time.

It was not until the late 19th century and early 20th century, with the advent of recording technology, that the scientific study of performance started (Binet and Courtier, 1896; Seashore, 1938). The second half of the 20th century saw an increased interest in looking at performance from the perspective of music psychology and cognition (Clynes, 1969; Gabrielsson, 1974; Longuet-Higgins and Lee, 1982, 1984; Clynes, 1986, 1987; Palmer, 1996). The field gained more attraction in the late 1980's, with advances in computers and electronic instruments, which facilitated more precise data capturing (Kirke and Miranda, 2013). These advances have allowed for empirical research that has focused on identifying factors that determine the way a musical piece is rendered. For example, aspects such as phrasing (Todd, 1992), meter (Sloboda, 1983), and intended emotions (Juslin, 2001), have been shown to have an effect on expressive variations in music performances.

Music performance science is a highly inter-disciplinary field, and a thorough review of the state of the art of the full field is outside the scope of this thesis. We refer the interested reader to the very comprehensive review articles by Palmer (1997) and Gabrielsson (1999, 2003). For a review of performance research from a musicological point of view see the edited volumes by Rink (1995, 2002).

1.1.2 Computational Methods of Expressive Performance

As stated above (or as the reader can guess from the title of this thesis), this work studies the use of (a family of data-driven) computational approaches to model expressive music performance. Computational models allow us to make experiments and test hypotheses related to certain cognitive aspects (Pylyshyn, 1984; Honing, 2006; Wiggins et al., 2012; Forth et al., 2016). An advantage of computational modeling over other modeling approaches is that computational models are *"open to direct and immediate test"* (Honing, 2006), and thus, are easier to evaluate and falsify.

Computational models can be used to study the way humans perform music. In this case, these models can be used to study characteristics of expressive performances by highlighting the relationships between properties of a musical piece and the way it is performed through changes in relevant dimensions such as dynamics (Kosta et al., 2016; Grachten et al., 2017; Cancino-Chacón et al., 2017d) and tempo (Friberg and Sundberg, 1999; Chew and Callender, 2013; Grachten and Cancino-Chacón, 2017). The study of music performance through computational approaches presents an opportunity to formalize certain aspects of what constitutes an expressive performance, such as expressive dynamics, tempo or articulation. While these aspects may seem very intuitive at first glance[3], research in computational models of performance has shown the challenges of translating these perceptual concepts into measurable properties that reflect the characteristics of a performance (Dixon et al., 2006; Chew and Callender, 2013; Elowsson and Friberg, 2017).

On the other hand, computational models of expressive performance can be used as tools for

[3] in particular to experienced musicians.

generating performances in their own right. These models can be used as standalone tools for automatically rendering expressive performances (Friberg et al., 2000; Kim et al., 2011; Cancino Chacón and Grachten, 2016), as interactive tools for allowing humans to shape a performance (Friberg, 2006; Baba et al., 2010; Canazza et al., 2015), or as accompaniment systems designed to perform music with humans and adapt to their performance styles (Raphael, 2001a; Xia et al., 2015; Cancino-Chacón et al., 2017a). In recent years, there has been an increased interest in developing systems that automatically generate (i.e. compose) music. Herremans et al. (2017) points out that automatic performance systems might be an important component in making automatic music generation usable by the general public.

But, is it possible to make an artificial system, a machine, that performs music like humans? In Stanislaw Lem's short story "Trurl's Electronic Bard", Trurl, the eponymous great machine constructor, attempts to create a machine capable of writing poetry. In order to build the machine, he *"collected eight hundred and twenty tons of books on cybernetics and twelve thousand tons of the finest poetry, then sat down to read it all. Whenever he felt he just couldn't take another chart or equation, he would switch over to verse, and vice versa. After a while it became clear to him that the construction of the machine itself was child's play in comparison with the writing of the program. The program found in the head of an average poet, after all, was written by the poet's civilization, and that civilization was in turn programmed by the civilization that preceded it, and so on to the very Dawn of Time, when those bits of information that concerned the poet-to-be were still swirling about in the primordial chaos of the cosmic deep. Hence in order to program a poetry machine, one would first have to repeat the entire Universe from the beginning—or at least a good piece of it."* (Lem, 2002, pp. 43–44). Do we really need to repeat (or simulate) the entire universe from the very beginning to program machine capable of generating expressive music performances? If we extend Lem's analogy to performing music expressively, this would imply that the "program" found in the head of the average musician is a product not only of human physiology, but also of the conventions and values of the society of the musician's era, which is in turn the product of the previous era (and society) and so on. Although described in a rather humorous way, Lem's quotations warns us of the incredible complexity of modeling an inherently (capital C) *Creative* activity such as music performance through a (necessarily) reductionistic approach.

As Widmer (2017) points out as two of the main theses of the Con Espressione Manifesto, *music is perceived by human listeners* and *music perception and appreciation are learned*. Therefore, it is important to note that, ultimately, music occurs in the *human mind* (Wiggins et al., 2010), i.e. the way we *experience* music is a product of active perception, and therefore, it would be naïve to think that any simplified (i.e. reductionistic) computational model can approach true levels of *artistic merit* or (more controversially) *creativity*, without considering a plethora of other factors, perhaps more importantly, *(embodied) human cognition* (Leman, 2008).

Whether building an artificial system capable of performing music like humans (i.e. *creatively*) is even possible is a philosophical question that, sadly, lies outside of the scope of this work. Nevertheless, it is important to point out that the efforts in the development of computational models of expressive performance are not without merit: a better understanding of musical expression is desirable in its own right, as fundamental scientific knowledge.

1.1.3 This Work

This thesis presents a comprehensive overview of the basis function models (BMs), a family of data-driven computational models of expressive music performance that have been developed over the past years, and that have been steadily growing in complexity. This model was originally

invented by Maarten Grachten and introduced in (Grachten and Widmer, 2012). Together with Maarten, I have been extending and improving this framework, and these extensions and improvements are what this thesis will be about.

The main motivation for this work is to study the complex relationship between the structural characteristics of a musical piece and the musically plausible ways that such a piece can be performed expressively.

1.2 Contributions

In the following chapters, I prefer to use the more neutral (and scientific) *we*, instead of the more selfish *I* (unless necessary by the limitations of my writing abilities), but since this chapter refers to the contributions presented in this work, it is easier to refer to them in first person singular. Still, it feels somehow awkward to refer to these contributions as mine without acknowledging the close collaboration and supervision of Maarten Grachten.

1. **A formalization of the BM framework for expressive performance.** Early publications on the BM framework lacked a rigorous formalization. In this thesis, I introduce a mathematical formalization of the components of the BM framework that highlights both their function, as well as their relation to each other.

2. **Non-linear and sequential BMs.** I expand upon the original linear formulation of the BM framework by using non-linear and sequential approaches, as well as for extending the framework from deterministic to probabilistic. Additionally, I introduce approaches for studying what the models are learning using variance-based sensitivity analysis, as well as differential sensitivity analysis.

3. **BM framework for ensemble performance.** This thesis presents an extension of the BM framework to model ensemble performances. This work was a collaboration with Thassilo Gadermaier and Maarten Grachten between the EU funded Lrn2Cre8[4] and PHENICX[5] projects. Thassilo and Maarten defined methods for aggregating the score information from different instruments. I designed and carried out the experiments, and contributed solutions to implementation issues such as the normalization of the features representing score information, as well as methods for analyzing the contributions of score features to the output of the model and comparing performances by different performers.

4. **The Basis Mixer**. In this work I present a detailed description of the Basis Mixer, an implementation of the BM framework for rendering expressive performances given an input score in MusicXML format. During the course of my doctoral degree, I collaborated with Maarten Grachten on this implementation, which builds on his previous work on musical expression. My contributions to the Basis Mixer include the implementation of the different computational models described in this work, including Bayesian linear models and artificial neural networks, the definition and implementation of some groups of descriptors of information in the score (i.e. basis functions) as well as optimizing the architectures of the models (including training them) for generating the performances.

5. **A prototype of an expressive accompaniment system.** As an application of models of expressive performance, I present a prototype of an accompaniment system that adapts to the expression of the (human) soloist. This work is part of an inter-institutional and

[4] http://lrn2cre8.eu.
[5] http://phenicx.com.

inter-project collaboration involving the Austrian Research Institute for Artificial Intelligence (OFAI), the Institute of Computational Perception (CP) at the Johannes Kepler University Linz (JKU) and the Department of Musical Acoustics (IWK) at the University of Music and Performing Arts Vienna (MDW) and the Con Espressione[6] and CoCreate[7] projects. The main components of the system were designed by Amaury Durand and myself, while the bulk of the implementation was done by Amaury, Martin Bonev and myself. Amaury designed and implemented the probabilistic score following under the supervision of Andreas Arzt. Martin and I adapted the first version of the prototype to work in real time for a MIDI input. I adapted the Basis Mixer for a real-time accompaniment scenario with occasional help from Maarten Grachten. Martin implemented the visualization under the supervision of Werner Goebl, Laura Bishop and myself.

6. **A review of the current state-of-the-art in computational models of expressive performance.** Finally, this works includes a critical and fairly comprehensive literature review of the current state of computational models of expressive performance that compares and positions the rest of the work presented in this thesis with current trends on the topic. This review also focuses on the missing elements and shortcomings of the current state of research, as well as presents some potential aspects to be focused on in future work.

1.3 Publications

The main content of this works builds on the following publications[8]:

1. Grachten, M., Cancino Chacón, C. E., and Widmer, G. (2014). Analysis and Prediction of Expressive Dynamics Using Bayesian Linear Models. In *Proceedings of the 1st International Workshop on Computer and Robotic Systems for Automatic Music Performance (SAMP 14)*, pages 545–552, Venice, Italy

2. Cancino Chacón, C. E. and Grachten, M. (2015). An Evaluation of Score Descriptors Combined with Non-linear Models of Expressive Dynamics in Music. In *Proceedings of the 18th International Conference on Discovery Science (DS 2015)*, pages 48–62, Banff, AB, Canada

3. Gadermaier, T., Grachten, M., and Cancino-Chacón, C. E. (2016). Basis-Function Modeling of Loudness Variations in Ensemble Performance. In *Proceedings of the 2nd International Conference on New Music Concepts (ICNMC 2016)*, Treviso, Italy

4. Cancino Chacón, C. E. and Grachten, M. (2016). The Basis Mixer: A Computational Romantic Pianist. In *Late Breaking/ Demo, 17th International Society for Music Information Retrieval Conference (ISMIR 2016)*, New York, NY, USA

5. Grachten, M. and Cancino-Chacón, C. E. (2017). Temporal dependencies in the expressive timing of classical piano performances. In Lessafre, M., Maes, P.-J., and Leman, M., editors, *The Routledge Companion to Embodied Music Interaction*, pages 360–369. Routledge

[6] http://www.cp.jku.at/research/projects/ConEspressione/.

[7] http://www.ofai.at/research/impml/projects/cocreate.html.

[8] The reader may notice that my surname appears sometimes as Cancino Chacón and sometimes as Cancino-Chacón in this list of publications. Unfamiliarity with Hispanic naming conventions has caused some of my publications to be incorrectly cited as *Chacón et al.* instead of *Cancino Chacón et al.* Therefore, in my publications from 2017 onwards, I decided to hyphen my surname, following a similar practice by other researchers of Hispanic ancestry.

6. Grachten, M., Cancino-Chacón, C. E., Gadermaier, T., and Widmer, G. (2017). Towards computer-assisted understanding of dynamics in symphonic music. *IEEE Multimedia*, 24(1):36–46

7. Cancino-Chacón, C. E., Gadermaier, T., Widmer, G., and Grachten, M. (2017d). An Evaluation of Linear and Non-linear Models of Expressive Dynamics in Classical Piano and Symphonic Music. *Machine Learning*, 106(6):887–909

8. Cancino-Chacón, C., Grachten, M., Sears, D. R. W., and Widmer, G. (2017c). What Were You Expecting? Using Expectancy Features to Predict Expressive Performances of Classical Piano Music. In *Proceedings of the 10th International Workshop on Machine Learning and Music (MML 2017)*, Barcelona, Spain

9. Cancino-Chacón, C., Bonev, M., Durand, A., Grachten, M., Arzt, A., Bishop, L., Goebl, W., and Widmer, G. (2017a). The ACCompanion v0.1: An Expressive Accompaniment System. In *Late Breaking/ Demo, 18th International Society for Music Information Retrieval Conference (ISMIR 2017)*, Suzhou, China

10. Cancino-Chacón, C. and Grachten, M. (2018). A Computational Study of the Role of Tonal Tension in Expressive Piano Performance. In *Proceedings of the 15th International Conference on Music Perception and Cognition (ICMPC15 ESCOM10)*, Graz, Austria

11. Cancino-Chacón, C., Grachten, M., Goebl, W., and Widmer, G. (2018). Computational Models of Expressive Music Performance: A Comprehensive and Critical Review. *To appear in Frontiers in Digital Humanities*

In addition to the before mentioned publications, I contributed to the following peer-reviewed publications as first author or as a co-author:

1. Cancino Chacón, C., Lattner, S., and Grachten, M. (2014a). Developing Tonal Perception Through Unsupervised Learning. In *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR 2014)*, pages 195–200

2. Lattner, S., Grachten, M., Agres, K., and Cancino Chacón, C. E. (2015b). Probabilistic Segmentation of Musical Sequences using Restricted Boltzmann Machines. In *Fifth International Conference on Mathematics and Computation in Music (MCM 2015)*, London, UK

3. Lattner, S., Cancino Chacón, C. E., and Grachten, M. (2015a). Pseudo-Supervised Training Improves Unsupervised Melody Segmentation. In *In Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)*, pages 2459–2465, Buenos Aires, Argentina

4. Agres, K., Cancino, C., Grachten, M., and Lattner, S. (2015). Harmonics co-occurrences bootstrap pitch and tonality perception in music: Evidence from a statistical unsupervised learning model. In *Proceedings of the Annual Meeting of the Cognitive Science Society (CogSci 2015)*, Pasadena, CA, USA

5. Velarde, Gissel and Weyde, Tillman and Cancino Chacón, Carlos and Meredith, David and Grachten, Maarten (2016). Composer Recognition Based On 2D-Filtered Piano Rolls. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR 2016)*, pages 116–121, New York, NY, USA

6. Cancino-Chacón, C., Grachten, M., and Agres, K. (2017b). From Bach to The Beatles: The Simulation of Human Tonal Expectation Using Ecologically-Trained Predictive Models. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR 2017)*, Suzhou, China

1.4 Thesis Outline

This section outlines the content of this thesis, and relates each chapter with its corresponding publications.

Chapter 2 presents an overview of computational models of expressive performance, including a brief description of computational strategies for modeling musical expression, as well as presenting a generic framework describing three main components of a computational model for expressive performance of notated music: the way the information in a musical score is represented by the model, the way the information in a musical performance is represented and a computational model relating these representations. Finally, this chapter also includes a description of the performance datasets used in the experiments presented in this thesis.

The main theoretical contributions of this thesis are contained in **Chapters 3 and 4**, which present a formalization and quantitative evaluation of the basis function models for musical expression. At this point I would like to point out that this thesis documents the historical development of the BM framework, so instead of simply describing the current state of the system, this work presents the evolution of the system starting from its original formulation introduced in (Grachten and Widmer, 2012). In this way, a significant part of this thesis is dedicated to describe the evolution of the score and performance representations, as well as the different computational models used to relate them. The purpose of these chapters is not mainly the historical overview of the BM framework in itself, but also that it is necessary for an adequate interpretation of experimental results throughout the years, since the experiments were realized using different variants of the BM framework.

Chapter 3 presents a mathematical formalization of the BM framework, and describes the linear version of the model both in its original deterministic formulation first proposed by Grachten and Widmer (2012) and a probabilistic version using Bayesian linear regression first proposed in (Grachten et al., 2014).

Chapter 4 continues with the formalization of the BM framework and describes two non-linear extensions of the framework, one non-sequential and another sequential, using artificial neural networks. This chapter introduces techniques for analyzing the contribution of the features representing score information by means of variance-based and differential sensitivity analysis. Parts of this chapter were previously published in (Cancino Chacón and Grachten, 2015), (Cancino-Chacón et al., 2017d) and (Grachten and Cancino-Chacón, 2017).

Chapter 5 presents an extension of the BM framework to allow for modeling ensemble performances. This chapter also describes the use of the BM framework as a tool for comparing expressive performances of symphonic music. This chapter is the only one that focuses on audio instead of MIDI performances. The content of this chapter consists mostly of material published in (Gadermaier et al., 2016) and (Grachten et al., 2017).

Chapter 6 explores the two kinds of cognitively inspired musical features: a set of features describing musical expectation, and a set of features describing harmonic tension, and the role that these features play in predicting expressive tempo and dynamics. This chapter consists of material published in (Cancino-Chacón et al., 2017c) and (Cancino-Chacón and Grachten, 2018).

Chapter 7 presents the Basis Mixer, an implementation of the BM framework to generate expressive performances given a musical score. This chapter consists of an extended version of Cancino Chacón and Grachten (2016).

Chapter 8 introduces an early prototype of the ACCompanion, an expressive accompaniment

system that uses a monophonic probabilistic score follower and an adapted version of the Basis Mixer to generate an accompaniment that adapts to the expression of the soloist. This chapter consists of an extended version of (Cancino-Chacón et al., 2017a).

Chapter 9 presents a comprehensive review of the state-of-the-art in computational models of expressive performances that takes into account the work presented in this thesis. This chapter serves as the conclusion of the thesis, and presents possible directions for future work. This chapter consists of a slightly edited version of (Cancino-Chacón et al., 2018), including its original introduction and motivation sections, so that it can be read as a self-contained document. Some parts of this introduction and Chapter 2 are recapitulated in this chapter for the sake of making it a standalone document.

After the main content of the thesis, I provide several appendices containing extra information relevant for building the models described in the main text, as well as unpublished preliminary experiments.

Appendix A contains a mathematical description of the full list of the score features (the so-called basis functions) described in the main text of this thesis.

Appendix B describes an method for computing expressive tempo that aims to be more perceptually plausible than the method described in the main text (in Chapters 3 and 4).

Appendix C describes preliminary work on joint estimation of expressive parameters. In this appendix we present an empirical analysis of the correlations of dimensions describing aspects of performance such as tempo, timing, dynamics and articulation in performances of Beethoven sonatas and Chopin piano music. We present preliminary results trying to determine whether there is an advantage if these dimensions describing expression are modeled jointly or independently.

Appendix D presents a probabilistic extension to the neural network models described in Chapter 4 using Gaussian mixture density networks (Bishop, 1994).

Appendices E and F contain details and derivations of methods for building and training the linear and non-linear models described in Chapters 3 and 4, respectively.

Appendix G presents an overview of methods for sampling expressive performances from probabilistic models.

Finally, **Appendix H** includes a list of mathematical formulae and identities used in this thesis.

As mentioned above, the main content of this work documents the historical development of the BM framework. The decision to take this approach instead of simply presenting the latest version of the framework allows for better storytelling, since it lets the reader understand the way that both experimental results and theoretical limitations of the models (e.g. the inability of earlier versions of the BM to capture the intrinsically sequential nature of music) motivated certain aspects of the development of the BM. Nevertheless, it is important to note that the empirical (quantitative) evaluation of this framework might not seem consistent across chapters: in some versions of the BM, I report experiments focused on expressive dynamics, in others versions I report experiments on expressive tempo, and so on. The reason for this (apparent) lack of consistency in the evaluation is two-fold:

1. The research conducted throughout my doctoral degree had a strong focus on understanding the way humans perform music (i.e. analyzing the way certain structural properties of the music contribute to aspects of expressive performance). In this case, it is more relevant (and methodologically easier) to test specific hypotheses on how certain structural aspects

of the score contribute to a specific aspect of performance. For example, an objective of an experiment described in Section 3.5[9] was testing whether describing gradual dynamics markings in a contextual fashion (i.e. considering combinations such as ***p*** → ***crescendo*** → ***f***) was more beneficial for modeling expressive dynamics than considering the individual markings without context.

2. Finally, for practical reasons: In some cases we focused on modeling a specific aspect of performance because we wanted to compare new improvements of the model with previous work focused on that aspect. For example, the experimental evaluation described in (Cancino Chacón and Grachten, 2015) (which is the basis for the experiments described in Section 4.2.3) focuses on expressive dynamics since it compares directly with the model of dynamics presented in (Grachten et al., 2014) (described in Section 3.5), which in turn compares directly with the experimental evaluation of the original linear BM introduced in (Grachten and Widmer, 2012). In another case, we received an invitation by the Lessafre et al. (2017), to contribute a chapter on expressive tempo for the Routledge Companion to Embodied Music Interaction, which resulted in (Grachten and Cancino-Chacón, 2017), which is the basis for the experiments is described in Section 4.3.5.

A companion website[10] contains an electronic version of this document, as well as links to code and examples of rendered performances and videos illustrating the models described in this thesis.

[9] Throughout this thesis, references to sections, equations and figures include the number of the chapter to which they belong. In the particular case of sections, Section $i.j.k$ will denote Chapter i, Section j, Subsection k.

[10] http://www.carloscancinochacon.com/documents/online_extras/phd_thesis/basis_function_models.html.

2 Computational Models of Expressive Performance

2.1 Introduction

In 1946 both Metro-Goldwyn-Mayer (MGM) and Warner Bros. (WB) released suspiciously similar short animated films starring anthropomorphized cartoon animals performing Franz Liszt's Hungarian Rhapsody No. 2 in C♯ minor, S. 244/2. Directed by William Hanna and Joseph Barbera, MGM's *The Cat Concerto* depicts Tom Cat performing the famous rhapsody, in front of a large and captivated audience, in spite of the mischievous Jerry Mouse, who (unsurprisingly) ends up stealing the spotlight. On the other hand, in *Rhapsody Rabbit*, WB's film directed by Fritz Freleng, Bugs Bunny performs the rhapsody only to be outsmarted by a small (and nameless) mouse. The similarity of the films sparked some controversy among the studios, after Technicolor, the company that developed the films for both studios, accidentally delivered footage from Rhapsody Rabbit to the MGM cartoon unit (Adamson, 1990, pp. 140–141). The Cat Concerto was awarded the 1947 Academy Award in the category of short subject (cartoon) (Academy of Motion Picture Arts and Sciences, 2015).

Although equally captivating, both performances of Liszt's rhapsody are very different from each other. Bugs Bunny's version, performed in real life by Polish-American pianist Jakob Gimpel, opens the piece with a lot of energy and *bravura*. On the other hand Tom Cat's version is more conservative, almost nostalgic. While the pianist "playing" Tom Cat is unknown, Gimpel's son speculates that it could have been the renowned Romantic virtuoso Shura Cherkassky (Gimpel, 2005). Furthermore, he states that *"Bugs's Rhapsody begins with a galvanizing burst of dramatic energy, whereas Tom's opening chords are somber and slow – very slow – an eccentric tempo my father would never have chosen"* (Gimpel, 2005).

We can use a computational approach to "visualize" the difference between these two performances, by comparing aspects relevant to expression, such as *tempo* and *dynamics*. We can represent these aspects using numerical parameters, which we will refer to in this work as *expressive parameters*, which encode characteristics of the performance. For example, we can represent expressive tempo by measuring the time between consecutive beats in beats per minute (bpm); and expressive dynamics by measuring the loudness of the performance, as the normalized energy of the audio signal. Figure 2.1 showcases the difference in these expressive parameters between both performances during the initial bars of the rhapsody[1]. The figure on the left shows the tempo curve in bpm, where we can see that Bugs's performance is overall faster, with a larger range (i.e. the "galvanizing burst of dramatic energy" described by Gimpel (2005)), whereas Tom's tempo is substantially slower and more restrained. The plot on the right shows the loudness curve for both performances. In this plot we can see that Tom's version is more restrained (i.e. the spread is smaller), whereas Bug's performance has a larger dynamic range and feels, therefore, more dramatic.

[1] The arrangement performed by Bugs Bunny makes several cuts to the original score, in particular bars 5-6 and 15-22. Figure 2.1 corresponds to the first 16 bars in Bugs Bunny's rhapsody, which correspond to the first 25 bars of the original score. In order to compare both versions, a manually edited version of Tom Cat's performance with the same cuts as Bugs Bunny's was generated.

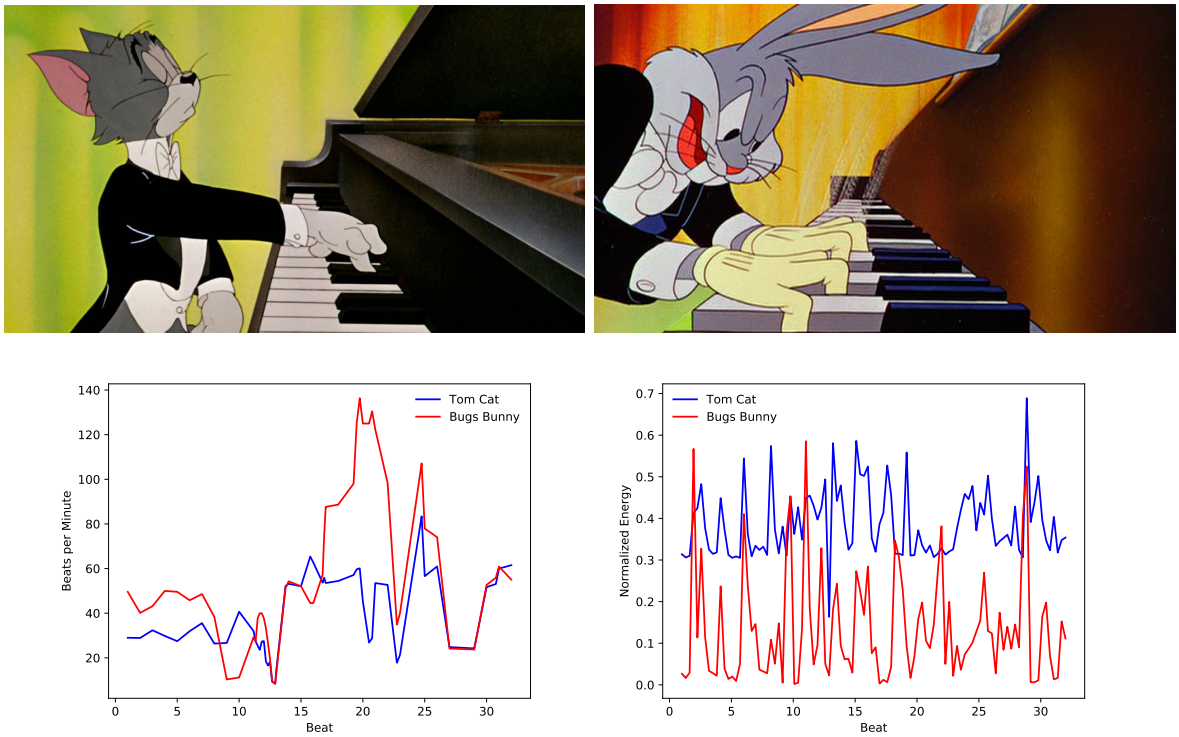


Figure 2.1: Tom Cat (top left) vs. Bugs Bunny (top right) performing Liszt's Hungarian Rhapsody No. 2. The plot in the bottom left compares the local tempo (in bpm) of both performances. The plot in the bottom right compares the performed dynamics (measured as the RMS energy of the signal normalized to lie in the interval $[0, 1]$). The images of Bugs Bunny and Tom Cat are property of Warner Bros. Entertainment Inc.

Comparing the curves to the score reveals that the moments of change in the expressive parameters tend to correspond to structural elements of the score. Figure 2.2 shows the excerpt of Liszt's rhapsody performed by Bugs Bunny aligned with his performed tempo (as a 1D heatmap). This comparison reveals that changes in expression tend to correspond to structural elements of the score such as the change in tempo in the **Lento a capriccio** section (bars 1-6), where the hue is mostly blue and violet and the **Andante, mesto** section (bars 7-16) where the hue is more green and yellow. Furthermore, we can see that local changes in tempo can be partially attributed to certain marks in the score like the decrease in tempo specified by ***rubato*** in bar 2, ***poco rit.*** in bar 4 and the ***piu ritenuto*** and fermatas in bar 6.

This example illustrates how computational methods can be used to analyze the way humans (or anthropomorphized cartoon characters) perform music. Computational models allow us to make experiments and test hypotheses related to certain cognitive aspects (Forth et al., 2016). These computational models can be then used to relate properties of the score (i.e. the composer) to the performer and the listener (Gingras et al., 2016).

In this chapter we focus on general aspects of computational models of expressive performance. The rest of this Chapter is structured as follows. Section 2.2 discusses some of the most used computational strategies for modeling musical expression. Section 2.3 describes the components of a computational model of performance. Finally, Section 2.4 describes the datasets used in this work.

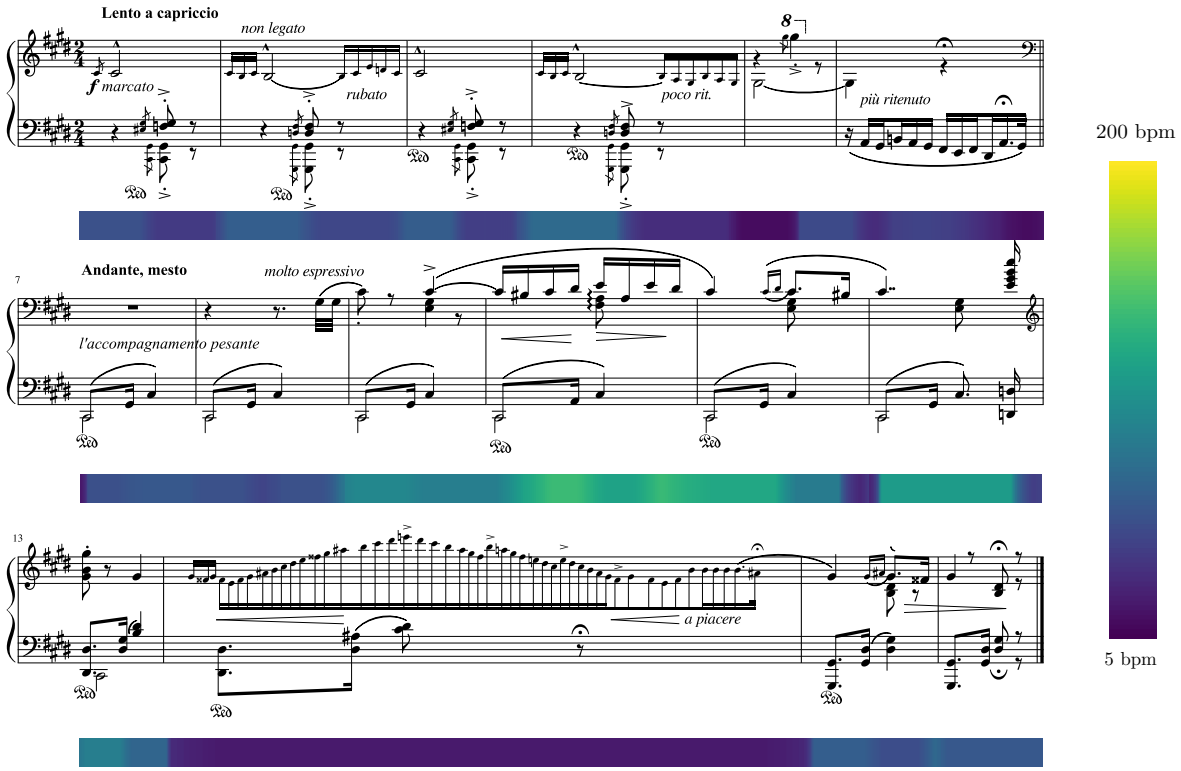


Figure 2.2: First 16 bars of the arrangement of Liszt's Rhapsody No. 2 as performed by Bugs Bunny (my transcription based on Edition Peters (Liszt, 1913)). The heatmap shows the tempo curve of Bug's performance (in red in the lower left plot in Figure 2.1).

2.2 Strategies for Modeling Musical Expression

While Chapter 9 presents an in-depth review of the current state-of-the-art in computational models of expressive performance (including the work presented in this thesis), the purpose of this Section is to present a very brief overview of some important concepts used throughout this work. For a more thorough review of strategies for modeling expressive performance, we refer the reader to the review papers by Widmer and Goebl (2004), De Poli (2004) and Kirke and Miranda (2013).

2.2.1 Rule-Based vs. Data-Driven Modeling

Very broadly speaking, we can identify two main approaches to modeling expressive performance through computational models, namely *rule-based* and *data-driven* approaches. Rule-based approaches attempt to define performance rules that relate aspects such as structural elements of the score, or expressive intentions of the user, to specific expressive performance actions. Examples of these rules include the increase in sound level proportional to pitch height (Sundberg et al., 1982) (e.g. perform higher notes louder), and the gradual slowing down at the end of a piece (Friberg and Sundberg, 1999). One of the most important examples of rule-based approaches is the KTH model (Sundberg et al., 1983; Friberg et al., 2000, 2006), which can be

thought of as a "musician's toolbox" (Friberg et al., 2006), i.e., a collection of methods that a musician might use to shape an expressive performance.

Rule-based models are useful because they can define robust models, since it is easy to control the influence of a particular rule in the performance. This kind of model can be used to test musically plausible hypotheses using an *analysis-by-synthesis* approach. On the other hand, it is unlikely that this approach produces models that can cover all possible performances (although they might cover a good part of the musically correct performances) (Friberg et al., 2006). Furthermore, it has been shown that rule based systems have some difficulty applying the rules in reverse, i.e. finding the rule set which would correspond to a given performance (Sundberg et al., 2003).

Data-driven approaches, on the other hand, attempt to infer performance rules directly from analyzing data from expressive performances. Therefore, this modeling approach is better suited for ecological analysis of performances (i.e. live concerts or commercial recordings, as opposed to performances conducted in *laboratory conditions*) than rule-based models. Early data-driven approaches (Widmer, 1995, 1996, 2000; Widmer and Tobudic, 2002; Widmer, 2003) aimed at learning explicit performance rules at various structural levels (from individual notes to higher phrasing levels), using methods like instance-based learning and inductive logic programming. Modern approaches (Widmer et al., 2009; Kim et al., 2011; Moulieras and Pachet, 2016; Cancino-Chacón et al., 2017d) tend to focus on machine learning techniques such as probabilistic graphical models and artificial neural networks, due to the development of more efficient learning algorithms and the increase of computational capacity. Still, a significant disadvantage of data-driven methods, in particular recent machine learning techniques, is that they require large amounts of data. Furthermore, while many data-driven models could theoretically describe all possible performances, in practice their capacity for explaining/predicting aspects of expressive performance is limited by assumptions encoded in their formulation. An example of these limitations might be the use of linear models that do not allow for interaction between different features (Cancino Chacón and Grachten, 2015), or models that only consider a small context to make predictions instead of the large scale temporal structures that are relevant in music (Widmer, 2017).

2.2.2 Analysis vs. Synthesis

Computational models of performance can be used for a wide variety of purposes, which can be broadly categorized into two groups: on one hand, these models can be used as an analytical tool for understanding the way humans perform music; on the other hand, we can use these models to synthesize (i.e. generate) new performances of musical pieces.

As analysis tools, computational models permit us to study the way humans perform music by investigating the relationship between certain aspects of the music, like the phrase structure, and aspects of expressive performance, such as expressive timing and dynamics. Furthermore, they allow us to investigate the close relationship between the roles of the composer, the performer and the listener (Kendall and Carterette, 1990; Gingras et al., 2016). Expressive performance and music perception form a feedback loop in which expressive performance actions (like a slowing down at the end of a phrase) are informed by perceptual expectations, and the perception of certain musical constructs (like grouping structure) is informed by the way the music is performed (Chew, 2016). In this way, computational models could also be used to enhance our understanding of the way humans listen to music.

On the other hand, computational performance models can be interesting in their own right, as tools for generating automatic or semi-automatic performances. In this case, a generative

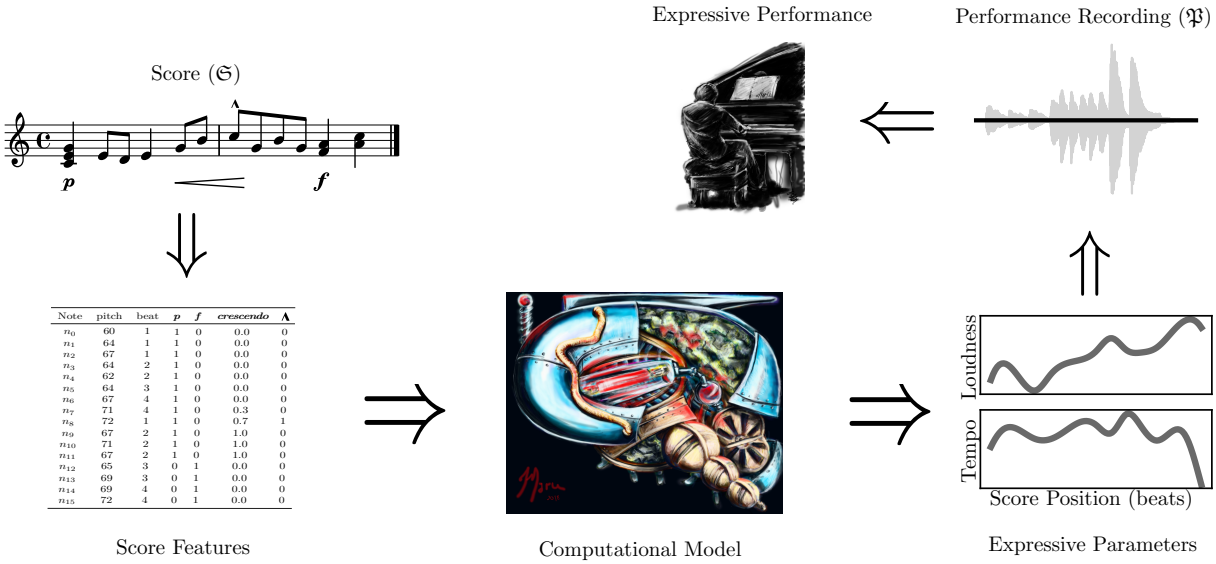


Figure 2.3: Framework for computational models of expressive performance.

system might attempt to produce a *convincing* or *human-like* performance of a piece of music given its score (Friberg et al., 2006; Grachten and Widmer, 2012; Okumura et al., 2014) or try to play alongside human musicians, not only tracking their expressive performance but also introducing its own expressive nuances (Xia et al., 2015; Cancino-Chacón et al., 2017a). Such systems might have many applications, including realistic playback in music typesetting tools (such as Finale or MuseScore) and automatic expressive accompaniment for rehearsing. Also, there is now a renewed interest in systems that automatically generate (i.e. compose) music. As pointed out by Herremans et al. (2017), automatic performance systems might be an important component in making automatic music generation usable by the general public.

2.3 Components of a Computational Model of Expressive Performance for Notated Music

In this Section we present a generic framework for describing research in expressive performance of notated music, which is shown in Figure 2.3. This framework consists of three elements, namely *score features*, which capture some of the information contained in the score, *expressive parameters*, which are numerical encodings of aspects of an expressive performance, and *computational models*, which relate them together. This framework can be understood as a simplified version of the framework proposed by Kirke and Miranda (2013).

It is important to note that until now, in this thesis we have used informal notions of a musical score and performance. While a comprehensive definition of what constitutes a musical score, or more importantly, what constitutes an expressive performance are philosophical questions that lie beyond the scope of the current work, in the following we present working definitions of these concepts:

Definition 2.1 (Musical Score)**.** A *musical score* (or simply *score*), denoted by $\mathfrak{S}$, is a symbolic representation of a musical composition. In this work, we consider a score to be a collection of elements that hold information about the music to be interpreted by a performer[2]. Given our

[2] Note that we intentionally do not define a score to be a mathematical *set* in the strictest sense, and indeed, a

focus on Western art music, it is relevant to distinguish between two kind of elements:

1. *note elements* (also referred to as *musical notes* or simply *notes*), i.e. elements that convey information about the note to be played, e.g. pitch (pitch class and octave), onset (the starting time of the note) and duration; and

2. *non-note elements*, which refer to elements such as rests, time and key signatures, or performance directives such as dynamics (e.g. ***p*** and ***f***), tempo (e.g. **Allegro** and **Andante**) or articulation markings (e.g. staccato, legato slurs).

The set of all possible scores is denoted as $\mathbb{S}$.

For computational purposes, in the rest of this work we will consider a musical score to be encoded in a machine-readable format such as MusicXML. While it could be argued that a deadpan MIDI counts as a score, such a format does not include other information that might be useful for the performer, such as enharmonic spelling, the time signature (and division in bars), dynamics and articulation markings, etc.

Definition 2.2 (Expressive Performance Recording)**.** A recording of an expressive performance, denoted as $\mathfrak{P}$, is a document that stores information in a physical or digital medium that encodes *measurable properties* of an expressive performance. These properties might include measurements of the sound waves generated by an instrument or loudspeaker (recorded as *audio*), the visual/motion information (recorded as video), abstract descriptions of parameters driving a digital instrument (such as the MIDI protocol), etc. $\mathbb{P}^{fmt}$ denotes the set of all expressive performances that can be captured using a format *fmt* (e.g. the set of all performances recorded in MIDI format is denoted as $\mathbb{P}^{\text{MIDI}}$).

While it is important to note that the recording is just a document that allows for storage (and possible reproduction) of the performance, not the performance itself, in the following we will refer to $\mathfrak{P}$ as the performance. For the purposes of this work, we will focus on performance recordings containing only MIDI or audio information (i.e. we are not considering the case of audio-visual information). If not otherwise stated, we assume that performances are in MIDI format, and thus, to unclutter notation we will simply write the set of all MIDI performances as $\mathbb{P}$.

2.3.1 Score Features

By *score features* – which are the inputs to the computational model – we denote descriptors used to represent a piece of notated music. Some of these features may be given directly by the score (such as notated pitches and durations), while others may be computed from the score in more or less elaborate ways, by some well-defined procedure.

We can roughly classify these features into three groups:

1. *low-level features*, i.e. features that can be readily extracted from the score without music theory (e.g. pitch, onset and duration of notes);

2. *mid-level features*, features that denote local level music theoretic concepts, such as encodings of the metrical stress patterns implied by a time signature, the scale degree of a note or the harmonic function of a chord (e.g. tonic, dominant); and

better way to describe a score would be as a data structure or an abstract data type. Nevertheless, we will abuse notation and write $x \in \mathfrak{S}$ to denote that x is an element of the score.

3. *high-level features*, features that denote advanced music theoretic concepts and abstract properties of the music, like intended emotion or aspects of the higher order structure of the piece (i.e. the relationship between different sections of the piece).

The process of extracting score features corresponds to the *Music/Analysis* module in Kirke and Miranda (2013)'s framework.

It is important to note that this definition does not necessarily require score features to be represented in a numerical form, and indeed many systems use features that are categorical. Still, recent advancements in machine learning have led to the use of numerical encodings of categorical information. Although there is a conceptual difference between these abstract non-numerical features and their numerical encodings, for the purpose of our discussion, this difference is not important. Therefore, in the rest of this work, we will assume that features can be encoded in a numerical fashion, and refer to these encodings as the features themselves.

2.3.2 Expressive Parameters

An *expressive parameter* – the output of a model – is a numerical encoding of an aspect of expressive performance. Here we would like to make a distinction between expressive parameters and the measurable properties of the performance from Definition 2.2. Although a computational model of performance might attempt to model expressive performances by directly generating audio signals (e.g. Raphael (2009)), most systems decompose the information of the performance into several aspects, most of which are motivated by different facets of our perception of music (which Raphael (2009) refers to as "consequences of the performance", rather than the performance itself). The description of an expressive performance by a set of parameters is necessarily a *theory-laden* activity[3], since the choice of parameters relies on knowledge, or assumptions on the nature of musical performance as a process.

In the case of (solo) piano music, the range of possible degrees of freedom for musical expression is limited by the mechanics of the instrument, compared to other acoustical instruments. Since most systems deal with piano music, the most common descriptors relate to loudness, expressive tempo and timing, and articulation (Kirke and Miranda, 2013; Widmer and Goebl, 2004), but of course they can also include other parameters like timbral features (Raphael, 2009; Ohishi et al., 2014).

The expressive parameters correspond to the outputs of the *Performance Knowledge* module in Kirke and Miranda (2013)'s framework. Section 9.4.1 presents a critical review of the choices involved in selecting and encoding these parameters.

2.3.3 Computational Models

A *computational model* then, in our context, is any computable function that maps score features to expressive parameters or, to be more precise, can make a *prediction* of the values of expressive parameters, given a score (represented via score features) as input. In music performance modeling, this is typically done by means of mathematical functions (probabilistic models, artificial neural networks, etc.) (Teramura et al., 2008; Kim et al., 2010; Grachten and Widmer, 2012) or by means of rules (Friberg et al., 2006; Canazza et al., 2015). Some of these models can be *trained* using a dataset of expressive performances. The model/function corresponds to the *Performance Knowledge* and the *Performance Context* in Kirke and Miranda (2013)'s

[3] According to philosophy of science, an observation is said to be theory-laden if it is affected by theoretical assumptions held by the researcher (Bogen, 2017).

Piece	Movements
Piano Sonata No. 1 in C Major, KV 279	1, 2, 3
Piano Sonata No. 2 in F Major, KV 280	1, 2, 3
Piano Sonata No. 3 in B♭ Major, KV 281	1, 2, 3
Piano Sonata No. 4 in E♭ Major, KV 282	1, 2, 3
Piano Sonata No. 5 in G Major, KV 283	1, 2, 3
Piano Sonata No. 6 in D Major, KV 284	1, 2, 3
Piano Sonata No. 10 in C Major, KV 330	1, 2, 3
Piano Sonata No. 11 in A Major, KV 331	1, 2, 3
Piano Sonata No. 12 in F Major, KV 332	1, 2, 3
Piano Sonata No. 13 in B♭ Major, KV 333	1, 2, 3
Piano Sonata No. 14 in C Minor, KV 457	1, 2, 3
Fantasia in C Minor, KV 475	1
Piano Sonata No. 15 in F Major, KV 53	1, 2, 3

Table 2.1: Musical material contained in the Batik/Mozart corpus

Piece	Movements
Piano Sonata No. 2 Op. 2 No. 2 in A Major	1, 2, 3, 4
Piano Sonata No. 8 Op. 13 in C Minor	1, 2, 3
Piano Sonata No. 9 Op. 14 No. 1 in E Major	1, 2, 3
Piano Sonata No. 12 Op. 26 in A♭ Major	1, 2, 3, 4
Piano Sonata No. 14 Op. 27 No. 2 in C♯ Major	1, 2, 3
Piano Sonata No. 17 Op. 31 No. 2 in D minor	1, 2, 3
Piano Sonata No. 21 Op. 53 in C Major	1, 2, 3
Piano Sonata No. 30 Op. 109 in E Major	1, 2, 3
Piano Sonata No. 32 Op. 110 in A♭ Major	1, 2, 3

Table 2.2: Musical material contained in the Zeilinger/Beethoven corpus

framework; the training of the model corresponds to the *Adaptation Process*, and the datasets are the *Performance Examples*.

2.4 Datasets Used in this Thesis

In order to study expressive performance of notated music using data-driven computational methods, it is necessary to use performance data that has been aligned to their scores. We can formalize this notion as follows:

Definition 2.3 (Matched Performance)**.** A *matched performance* is a triple $\mathfrak{P}_{\mathfrak{S}} = (\mathfrak{P}, \mathfrak{S}, \text{match})$ consisting of an expressive performance $\mathfrak{P}$, its corresponding score $\mathfrak{S}$ and match: $\mathfrak{P} \mapsto \mathfrak{S}$, a map that assigns events in the performance (e.g. a performed MIDI note or a position in time in an audio recording) to their corresponding elements in the score. A performance is said to be

1. *matched note-wise* if there is a mapping assigning performed notes (as the case of MIDI recordings) to the corresponding score notes; or
2. *matched onset-wise* if there is a mapping between performed onset times and their corresponding temporal positions in the score, without matching a performance event to individual notes at each position.

The set of all matched performances of score $\mathfrak{S}$ is denoted as $\mathbb{P}_{\mathfrak{S}}$.

In this section we describe 4 datasets of expressive performances aligned to their scores used

Composer	Piece	Movements	Conductor
Beethoven	Symphony No. 5 Op. 67 in C Minor	1, 2, 3, 4	Fischer
Beethoven	Symphony No. 6 Op. 68 in F Major	1, 2, 3, 4, 5	Fischer
Beethoven	Symphony No. 9 Op. 125 in D Minor	1, 2, 3, 4	Fischer
Mahler	Symphony No. 4 in G Major	1, 2, 3, 4	Jansons
Bruckner	Symphony No.9 WAB 109 in D Minor	1, 2, 3	Jansons

Table 2.3: Musical material contained in the RCO/Symphonic corpus

throughout this work. Each of these datasets corresponds to a single performer/single orchestra. The first three datasets consist of solo performances of classical piano music recorded on computer controlled grand pianos. The musical material contained in these datasets includes music from the second half of the 18th century to the end of the 19th century, comprising the Classical and Romantic periods.

2.4.1 Batik/Mozart

The Batik/Mozart corpus is a note-wise matched dataset which consists of recordings by Austrian pianist Roland Batik (b. 1951) of 12 piano sonatas and the Fantasia in C Minor KV 475, performed on a computer controlled Bösendorfer SE 290 (Widmer and Tobudic, 2002). The dataset contains 37 pieces (each movement of a sonata is considered an individual piece), which result in ca. 100 000 performed notes, for nearly 4 hours of music. The pieces in this corpus are shown in Table 2.1. An important characteristic of this dataset is that the melody was manually annotated for all of the pieces, allowing us to model characteristics of the melody and harmony separately.

The performances have been aligned to their corresponding machine-readable scores in an in-house format (called *matchfile*), developed by Gerhard Widmer and his colleagues. This dataset follows the Henle Verlag[4] edition of the Mozart sonatas, although information about dynamics or articulation markings was not included in the dataset.

2.4.2 Magaloff/Chopin

The Magaloff/Chopin corpus (Flossmann et al., 2010) is a note-wise matched dataset which consists of the complete Chopin piano solo works (excluding a few posthumously published works) performed by renowned pianist Nikita Magaloff (1912-1992) during a series of concerts in Vienna, Austria in 1989. These performances were recorded using a Bösendorfer SE computer-monitored grand piano, and then converted into standard MIDI format.

The performances have been aligned to their corresponding machine-readable musical scores in MusicXML format, which were obtained from hard-copies of the sheet music using commercial optical music recognition software[5] and subsequent manual correction. We have used the Henle Urtext Edition wherever possible, which explicitly states its intention to stay faithful to Chopin's original manuscripts. The score–performance alignment step has also been performed semi-automatically, involving manual correction of automatic alignments.

The data set comprises more than 150 pieces and over 300 000 performed notes, adding up to almost 10 hours of music. The pieces in this dataset are shown in Table 2.4.

[4] https://www.henle.de
[5] http://www.visiv.co.uk/

2.4.3 Zeilinger/Beethoven

This is another note-wise matched corpus of classical piano performances, very similar in form to the Magaloff/Chopin corpus. It consists of the performances of 29 different movements from 9 different Beethoven Sonatas by Austrian concert pianist Clemens Zeilinger, recorded under studio conditions at the Anton Bruckner University in Linz (Austria), on January 3-5, 2013. The pieces were performed on a Bösendorfer CEUS 290 computer-monitored grand piano, and converted to standard MIDI format. Further preparation of the data, such as the production of machine-readable scores (using the Henle Urtext Edition), and score to performance alignment were done in the same way as for the Magaloff/Chopin corpus. This data set comprises over 70 000 performed notes, adding up to just over 3 hours of music. The pieces in this dataset are shown in Table 2.2.

It is important to note that the process of obtaining machine-readable scores through commercial optical music recognition is not a perfect process and there might be some mistakes in the data. Therefore, the first round of manual corrections for the Magaloff/Chopin and Zeilinger/Beethoven datasets were performed by Sebastian Flossmann and colleagues. To the date of this writing, the last round of corrections for dynamics and timing markings for the Magaloff/Chopin and Zeilinger/Beethoven corpora was performed during November 2016 to September 2017 by Maarten Grachten and myself.

2.4.4 RCO/Symphonic

The RCO/Symphonic corpus is an onset-wise matched dataset consisting of symphonies from the Classical and Romantic periods. It contains recorded performances (audio), machine-readable representations of the musical score (MusicXML) and automatically produced (using the method described in Grachten et al. (2013)), manually corrected alignments between score and performance, for each of the symphonies. The manual corrections were made either at bar or at beat level (depending on the tempo of the piece), and subsequently, the performance was re-aligned automatically, using the corrected positions as anchors.

The pieces were performed by the Royal Concertgebouw Orchestra (RCO), conducted by either Iván Fischer or Mariss Jansons, at the Concertgebouw in Amsterdam, the Netherlands. The corpus amounts to a total of 20 movements from 5 works, listed in Table 2.3. The corresponding performances sum up to a total length of over 4.5 hours of music. The 20 scores result in ca. 54 000 note onsets.

The symbolic scores (in MusicXML format) are provided partly by Bärenreiter Verlag[6], and partly by Donemus Publishing[7].

The loudness of the recordings was computed using the EBU R 128 loudness measure (EBU-R-128, 2011) which is the recommended way of comparing loudness levels of audio content in the broadcasting industry. This measure takes into account human perception, particularly the fact that signals of equal power but different frequency content are not perceived as being equally loud. To obtain instantaneous loudness values, we compute the measure on consecutive blocks of audio, using a block size and hop size of 1024 samples, using a 44 100 Hz sample rate.

Through the score–performance alignment, the resulting loudness curve is indexed by musical time (such that we know the instantaneous loudness of the recording at, say, the second beat of measure 64 in the piece), and is thus associated to the score representation of the piece.

[6] http://www.baerenreiter.com
[7] http://www.donemus.nl

Pieces		
Rondo Op. 1	Etude Op. 25 No. 7	Nocturne Op. 37 No. 1
Piano Sonata No. 1 Op. 4 (3 mvts.)	Etude Op. 25 No. 8	Nocturne Op. 37 No. 2
Rondo Op. 5	Etude Op. 25 No. 9	Ballade Op. 38
Mazurka Op. 6 No. 1	Etude Op. 25 No. 10	Scherzo Op. 39
Mazurka Op. 6 No. 2	Etude Op. 25 No. 11	Polonaises Op. 40 No. 1
Mazurka Op. 6 No. 3	Polonaises Op. 26 No. 1	Polonaises Op. 40 No. 2
Mazurka Op. 6 No. 4	Polonaises Op. 26 No. 2	Mazurka Op. 41 No. 1
Mazurka Op. 7 No. 1	Nocturne Op. 27 No. 1	Mazurka Op. 41 No. 2
Mazurka Op. 7 No. 2	Nocturne Op. 27 No. 2	Mazurka Op. 41 No. 3
Mazurka Op. 7 No. 3	Prelude Op. 28 No. 1	Mazurka Op. 41 No. 4
Mazurka Op. 7 No. 4	Prelude Op. 28 No. 2	Waltz Op. 42
Mazurka Op. 7 No. 5	Prelude Op. 28 No. 3	Tarantelle Op. 43
Nocturne Op. 9 No. 1	Prelude Op. 28 No. 4	Polonaises Op. 44
Nocturne Op. 9 No. 2	Prelude Op. 28 No. 5	Prelude Op. 45
Nocturne Op. 9 No. 3	Prelude Op. 28 No. 6	Allegro de Concert Op. 46
Etude Op. 10 No. 1	Prelude Op. 28 No. 7	Ballade Op. 47
Etude Op. 10 No. 2	Prelude Op. 28 No. 8	Nocturne Op. 48 No. 1
Etude Op. 10 No. 3	Prelude Op. 28 No. 9	Nocturne Op. 48 No. 2
Etude Op. 10 No. 4	Prelude Op. 28 No. 10	Fantaisie Op. 49
Etude Op. 10 No. 5	Prelude Op. 28 No. 11	Mazurka Op. 50 No. 1
Etude Op. 10 No. 6	Prelude Op. 28 No. 12	Mazurka Op. 50 No. 2
Etude Op. 10 No. 7	Prelude Op. 28 No. 13	Mazurka Op. 50 No. 3
Etude Op. 10 No. 8	Prelude Op. 28 No. 14	Impromptu Op. 51
Etude Op. 10 No. 9	Prelude Op. 28 No. 15	Ballade Op. 52
Etude Op. 10 No. 10	Prelude Op. 28 No. 16	Polonaises Op. 53
Etude Op. 10 No. 11	Prelude Op. 28 No. 17	Scherzo Op. 54
Etude Op. 10 No. 12	Prelude Op. 28 No. 18	Nocturne Op. 55 No. 1
Variations brillantes Op. 12	Prelude Op. 28 No. 19	Nocturne Op. 55 No. 2
Nocturne Op. 15 No. 1	Prelude Op. 28 No. 20	Mazurka Op. 56 No. 1
Nocturne Op. 15 No. 2	Prelude Op. 28 No. 21	Mazurka Op. 56 No. 2
Nocturne Op. 15 No. 3	Prelude Op. 28 No. 22	Mazurka Op. 56 No. 3
Rondo Op. 16	Prelude Op. 28 No. 23	Berceuse Op. 57
Mazurka Op. 17 No. 1	Prelude Op. 28 No. 24	Piano Sonata No. 3 Op. 58 (3 mvts.)
Mazurka Op. 17 No. 2	Impromptu Op. 29	Mazurka Op. 59 No. 1
Mazurka Op. 17 No. 3	Mazurka Op. 30 No. 1	Mazurka Op. 59 No. 2
Mazurka Op. 17 No. 4	Mazurka Op. 30 No. 2	Mazurka Op. 59 No. 3
Waltz Op. 18	Mazurka Op. 30 No. 3	Barcarolle Op. 60
Bolero Op. 19	Mazurka Op. 30 No. 4	Polonaises Op. 61
Scherzo Op. 20	Scherzo Op. 31	Nocturne Op. 62 No. 1
Ballade Op. 23	Nocturne Op. 32 No. 1	Nocturne Op. 62 No. 2
Mazurka Op. 24 No. 1	Nocturne Op. 32 No. 2	Mazurka Op. 63 No. 1
Mazurka Op. 24 No. 2	Mazurka Op. 33 No. 1	Mazurka Op. 63 No. 2
Mazurka Op. 24 No. 3	Mazurka Op. 33 No. 2	Mazurka Op. 63 No. 3
Mazurka Op. 24 No. 4	Mazurka Op. 33 No. 3	Waltz Op. 64 No. 1
Etude Op. 25 No. 1	Mazurka Op. 33 No. 4	Waltz Op. 64 No. 2
Etude Op. 25 No. 2	Waltz Op. 34 No. 1	Waltz Op. 64 No. 3
Etude Op. 25 No. 3	Waltz Op. 34 No. 2	Waltz Op. 69 No. 1
Etude Op. 25 No. 4	Waltz Op. 34 No. 3	Nocturne Op. 72 No. 1
Etude Op. 25 No. 5	Piano Sonata No. 2 Op. 35 (3 mvts.)	
Etude Op. 25 No. 6	Impromptu Op. 36	

Table 2.4: Musical material contained in the Magaloff/Chopin corpus

3 Basis Function Models I: Linear Models

This chapter contains material published in

- Grachten, M., Cancino Chacón, C. E., and Widmer, G. (2014). Analysis and Prediction of Expressive Dynamics Using Bayesian Linear Models. In *Proceedings of the 1st International Workshop on Computer and Robotic Systems for Automatic Music Performance (SAMP 14)*, pages 545–552, Venice, Italy

3.1 Introduction

In Chapter 2 we discussed general strategies for studying musical expression using computational models. The following two chapters will describe the *basis function models* (BMs), a data-driven approach for modeling musical expression that formalizes the components of the generic framework discussed in Section 2.3 using a machine learning paradigm.

Broadly speaking, the BM framework uses *basis functions*, i.e. numerical encodings of a variety of descriptors of a musical score (score features), as inputs to a mathematical function (which we call *predictive function*) that relates them to the expressive targets, i.e. measurable characteristics of aspects of an expressive performance.

Work on the BM framework is ongoing, so instead of simply describing the most recent iteration of the BM, the following two chapters document the development of the BMs since their original linear versions (this chapter) to the more recent non-linear and sequential versions (Chapter 4). The motivation behind this structure is two-fold: on the one hand, it allows us to present each variant of the model as a stepping stone, with advantages and disadvantages; on the other hand, it provides a better storytelling, by allowing us to focus on the goals and the motivations for each iteration of the model (including the things that did not work), and thus, provides an insight into the development of complex models starting from simple premises.

It is important to note that the BM approach is a generic framework to model musical expression, which could be used to describe other machine learning models for expressive performance like those proposed by Teramura et al. (2008), Kim et al. (2011) and Okumura et al. (2014).

The rest of this chapter is structured as follows: In Section 3.2 we provide a general definition of the BM framework by formalizing the three components of a model of expressive performance described in Section 2.3. Section 3.3 presents an overview of the first linear formulation of the BM. Section 3.4 describes a probabilistic version of the linear model using a Bayesian model. Finally, Section 3.7 concludes this chapter.

3.2 The Basis Function Models: A Formal Description

This section provides a description of the BM framework that formalizes the concepts of expressive parameters, score features and computational models, using a mathematical description.

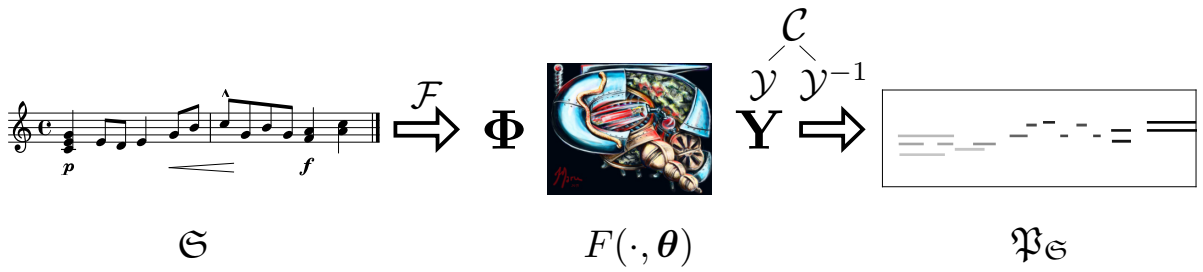


Figure 3.1: A basis function model for modeling expressive performances: Aspects of a score $\mathfrak{S}$ are represented numerically by $\mathbf{\Phi}$, using a score representation model $\mathcal{F}$. Aspects of an expressive performance are captured using a performance codec $\mathcal{C}$, consisting of a performance representation model $\mathcal{Y}$ and a performance decoder $\mathcal{Y}^{-1}$. The score representation $\mathbf{\Phi}$ serves as input to a predictive function $F(\cdot, \boldsymbol{\theta})$, which outputs $\mathbf{Y}$, a prediction of a performance representation under representation model $\mathcal{Y}$. Using a performance decoder $\mathcal{Y}^{-1}$, we can generate a matched performance $\mathfrak{P}_{\mathfrak{S}}$. $\mathbf{\Phi}$, $\mathbf{Y}$ and F correspond to the score features, expressive parameters and computational model described in Section 2.3. See text for a detailed explanation of these concepts.

Figure 3.1 shows a schematic representation of the BM framework and its components, which will be described in detail in this section.

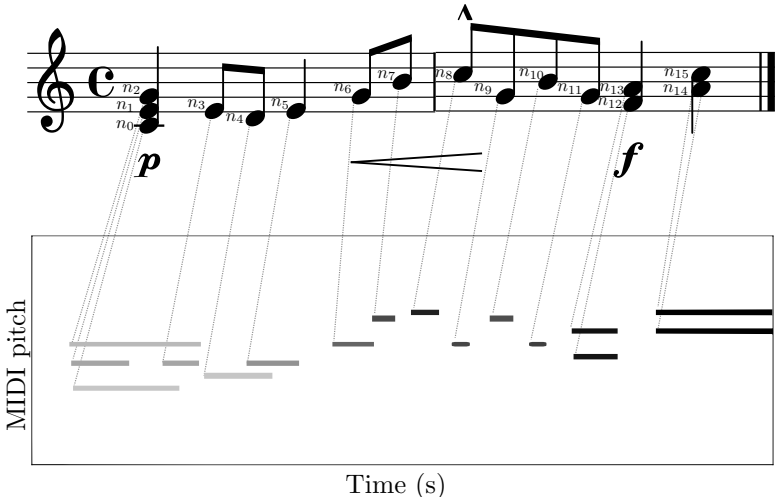
Following the notation introduced in Section 2.3, $\mathfrak{P}_{\mathfrak{S}}$ denotes an expressive performance matched to score $\mathfrak{S}$ and $\mathbb{P}_{\mathfrak{S}}$ is the set of all matched performances of score $\mathfrak{S}$. In the rest of this work, $\boldsymbol{\mathcal{X}} = \{n_0, \ldots, n_{N_x-1}\}$ represents the set of $|\boldsymbol{\mathcal{X}}| = N_x$ notes in a musical score and $\boldsymbol{\mathcal{O}} = \{o_0, \ldots, o_{N_o-1}\}$ represents the set of $|\boldsymbol{\mathcal{O}}| = N_o$ unique score positions, i.e. the temporal position of musical events, commonly measured in beats, which will be referred to as *score onsets*[1]. The set of all notes that occur at the same score position as score note n_i will be denoted as $o(n_i) = \{n_j \in o \mid \text{onset}(n_j) = \text{onset}(n_i)\}$, with $\text{onset}(n_i)$ being the onset time of n_i. An arbitrary element of the score (note or onset) will be denoted as x. See Figure 3.4 for an example of a musical score and its different elements.

3.2.1 Expressive Parameters

Let us consider the matched performance shown in Figure 3.2. In this figure, the plot on the left represents the matching between performed MIDI notes, represented as a piano roll, and their corresponding notes in the musical score. In this piano roll, a darker hue means a louder note. On the right side of this plot, we have three measurable quantities describing the performance information in MIDI format, namely, the performed MIDI velocity, onset time and duration of each note in the score[2]. The MIDI velocity is an indication of the loudness of a note (a higher MIDI velocity means a louder note). The performed onset time describes the point in time at which the note was performed (i.e. the time at which a MIDI device sends a `NOTE ON` message). In standard MIDI format, this is reported in multiples of a minimal unit of time called MIDI tick, but for convenience in the example shown in Figure 3.2 it is shown in seconds. Note that in a piano performance, defining the duration of a note as simply the time interval between its `NOTE ON` and a `NOTE OFF` messages without considering pedal information might not be musically meaningful. To estimate the sounding duration of a note in a piano performance

[1] It is easy to see that $N_x \geq N_o$, with the equality occurring only in the case of monophonic music.

[2] The MIDI format allows for other parameters describing a performance such as pitch bending and sustain pedal.



Note	vel(·)	onset$_{perf}$(·)	duration$_{perf}$(·)
n_0	28	0.24	0.60
n_1	44	0.23	0.33
n_2	34	0.21	0.75
n_3	44	0.73	0.21
n_4	28	0.96	0.39
n_5	53	1.21	0.30
n_6	75	1.69	0.23
n_7	88	1.90	0.14
n_8	108	2.12	0.16
n_9	90	2.36	0.10
n_{10}	86	2.56	0.14
n_{11}	93	2.79	0.09
n_{12}	114	3.03	0.25
n_{13}	116	3.03	0.26
n_{14}	123	3.49	0.67
n_{15}	124	3.49	0.66

Figure 3.2: Piano roll of performance matched onsetwise $\mathfrak{P}_{\mathfrak{S}}$(left) and information describing the performance for each note (right). The intensity of the hue of the notes on the piano roll represents their MIDI velocity. Functions vel(·), onset$_{perf}$(·) and duration$_{perf}$(·) represent the performed MIDI velocity, onset time and duration (in seconds) of each note in the score $\mathfrak{S}$.

it is necessary to include information from the sustain pedal. In the rest of this discussion we will simply assume that a method for estimating the sounding duration of a note exists, and the term *duration* will refer to this sounding duration[3].

From the representation of a performance in MIDI format, we can see that a straightforward way to represent expressive dynamics is to consider directly the MIDI velocity[4]. We represent the MIDI velocity of the i-th note in the score as vel(n_i). In this case, we can consider vel(·) as a function which maps notes in the score $\mathfrak{S}$ to their performed MIDI velocity in the matched performance $\mathfrak{P}_{\mathfrak{S}}$. Other aspects of a performance like expressive tempo might not be immediately extracted from the MIDI format, but have to be calculated considering information from the score. For example, we can define a function bpm(·) that computes the local tempo in bpm at each score position considering the performed onset times (in seconds) of each note (see the column onset$_{perf}$(·) in Figure 3.2) and the score position of each note in the score (in beats)[5]. We can generalize this idea of representing aspects of an expressive performance through functions that map elements of a score (e.g. notes in the example above) to their corresponding property in the matched performance (their MIDI velocity in the above example) as follows:

Definition 3.1 (Expressive Encoding Function)**.** An *expressive encoding function* is a function $\mathsf{y}\colon \mathfrak{P}_{\mathfrak{S}} \mapsto \mathbb{R}$ that encodes an event in a matched expressive performance (e.g. a performed MIDI note or a particular position in time in an audio recording) into an *expressive parameter*, a numerical encoding of an aspect of an expressive performance. The value of such an expressive parameter[6] corresponding to score element $x_i \in \mathfrak{S}$ is denoted as $\mathsf{y}(x_i \mid \mathfrak{P}_{\mathfrak{S}}) = t_i$ with $t_i \in \mathbb{R}$.

[3] This working definition of duration of a note is a simplification. A more general concept of sounding duration of a note should involve the acoustic properties of the instrument and the room in which the note is performed.

[4] It is important to note that the MIDI velocity is by no means a perfect indicator of loudness or dynamics, since it depends on the instrument in which it was performed.

[5] In this case, bpm(·) returns the same value for all notes in the same score position, as would be the case of the notes in the C major chord at the beginning of the score in Figure 3.2, i.e. bpm(n_0) = bpm(n_1) = bpm(n_2).

[6] Borrowing from the terminology of the machine learning literature, we select t to represent the value of expressive parameters since they are the values that we are ultimately interested in predicting, i.e. the *target values*.

We slightly abuse notation, and denote the values of an expressive parameter for all *relevant* elements[7] in $\mathfrak{S}$ as $\mathsf{y}(\mathfrak{S} \mid \mathfrak{P}_{\mathfrak{S}}) = \mathbf{t}$, with $\mathbf{t} \in \mathbb{R}^N$.

In the example above, we can identify $\mathsf{y}_{\text{vel}}(n_i \mid \mathfrak{P}_{\mathfrak{S}}) = \text{vel}(n_i)$ and $\mathsf{y}_{\text{bpm}}(n_i \mid \mathfrak{P}_{\mathfrak{S}}) = \text{bpm}(n_i)$ as expressive encoding functions relating score note n_i to its performed MIDI velocity and local tempo in bpm in performance $\mathfrak{P}_{\mathfrak{S}}$, respectively. In this case, MIDI velocity is an example of an expressive parameter describing expressive dynamics, while bpm is an example of an expressive parameter describing expressive tempo. This example shows how in order to describe several aspects of an expressive performance, such as expressive tempo, dynamics and articulation, we might need to use different expressive parameters, each represented by different expressive encoding functions. We can represent these different aspects of expressive performance describing elements in the score as points in a multidimensional space, where each dimension represents an expressive parameter. We can formalize this notion as follows:

Definition 3.2 (Performance Space, Performance Representation)**.**

1. A *performance representation model* is a set of N_y expressive encoding functions $\mathcal{Y} = \{\mathsf{y}_1, \ldots, \mathsf{y}_{N_y}\}$ covering several aspects of an expressive performance. Using a slight abuse of notation we will also write $\mathcal{Y}(x_i \mid \mathfrak{P}_{\mathfrak{S}}) = \big(\mathsf{y}_1(x_i \mid \mathfrak{P}_{\mathfrak{S}}), \ldots, \mathsf{y}_{N_y}(x_i \mid \mathfrak{P}_{\mathfrak{S}})\big)^{\mathrm{T}} = \boldsymbol{t}_i$, with $\boldsymbol{t}_t \in \mathbb{R}^{N_y}$, as a vector valued function $\mathcal{Y}\colon \mathfrak{P}_{\mathfrak{S}} \mapsto \mathcal{P}$ which maps the aspects of the performance captured by the expressive parameters into a vector subspace $\mathcal{P} \subset \mathbb{R}^{N_y}$. We refer to $\mathcal{P}$ as the *performance space* spanned by the expressive parameters. In this case, each expressive parameter represents a coordinate in the performance space (i.e. a component of vector function $\mathcal{Y}(x_i, \mathfrak{P}_{\mathfrak{S}})$), and we will refer to the j-th expressive parameter as the output of expressive encoding function y_j. Nevertheless, for the sake of clarity, whenever we are referring to a specific parameter we will write $\mathsf{y}_{\text{name of the parameter}}$ instead of simply writing $\mathsf{y}_{\textit{parameter index}}$, as done in the example above with y_{vel} and y_{bpm} instead of y_1 and y_2.
2. The *performance representation of* $\mathfrak{P}_{\mathfrak{S}}$ *under* $\mathcal{Y}$ (or simply *performance representation of* $\mathfrak{P}_{\mathfrak{S}}$), denoted as $\mathcal{Y}(\mathfrak{S} \mid \mathfrak{P}_{\mathfrak{S}}) = \mathbf{T}$, is the value of the expressive parameters represented by $\mathcal{Y}$ for all relevant elements of $\mathfrak{S}$. In most cases $\mathbf{T}$ will be a real valued matrix in $\mathbb{R}^{N \times N_y}$, where N is the number of relevant elements of $\mathfrak{S}$ and its entry in row i and column j, denoted by t_{ij}, is given by $\mathsf{y}_j(x_i \mid \mathfrak{P}_{\mathfrak{S}})$, i.e. the value of the j-th expressive parameter for score element x_i. In this work (in particular in Chapter 4), we will explore cases where some expressive encoding functions are defined note-wise, i.e. at the note level, (e.g. the MIDI velocity of individual notes) while others are defined onset-wise, i.e. at the onset level, (e.g. the local tempo at a given score position), where $\mathbf{T} = \{\mathbf{T}_{\text{onset-wise}}, \mathbf{T}_{\text{note-wise}}\}$ is a composite structure, with $\mathbf{T}_{\text{onset-wise}} \in \mathbb{R}^{N_{No} \times N_{y_{\text{ow}}}}$ and $\mathbf{T}_{\text{note-wise}} \in \mathbb{R}^{N_{N_x} \times N_{y_{\text{nw}}}}$, where $N_{y_{\text{ow}}}$ and $N_{y_{\text{nw}}}$ are the number of expressive encoding functions defined onset-wise and note-wise, respectively.

In the example above, we have $\mathcal{Y} = \{\mathsf{y}_{\text{vel}}, \mathsf{y}_{\text{bpm}}\}$, which results in performance space $\mathcal{P}$ being a 2D space, shown in Figure 3.3. In this plot, each point corresponds to $\mathcal{Y}(n_i \mid \mathfrak{P}_{\mathfrak{S}})$, i.e. the representation of the performance of score note n_i in performance $\mathfrak{P}_{\mathfrak{S}}$.

A performance representation model $\mathcal{Y}$ indicates the way a computational model of expressive performance describes (i.e. encodes) characteristics of the expressive performance. However, for a model to be able to generate a performance rendering of a piece given its score, an additional

[7] By relevant elements of $\mathfrak{S}$ we mean those score elements (e.g. notes or onsets) for which the values of the expressive parameters are defined. For example, it might not be meaningful to define the value of the local tempo in bpm for all ***piano*** markings, but it makes sense to define it for a specific score position (e.g. the first beat in the third bar).

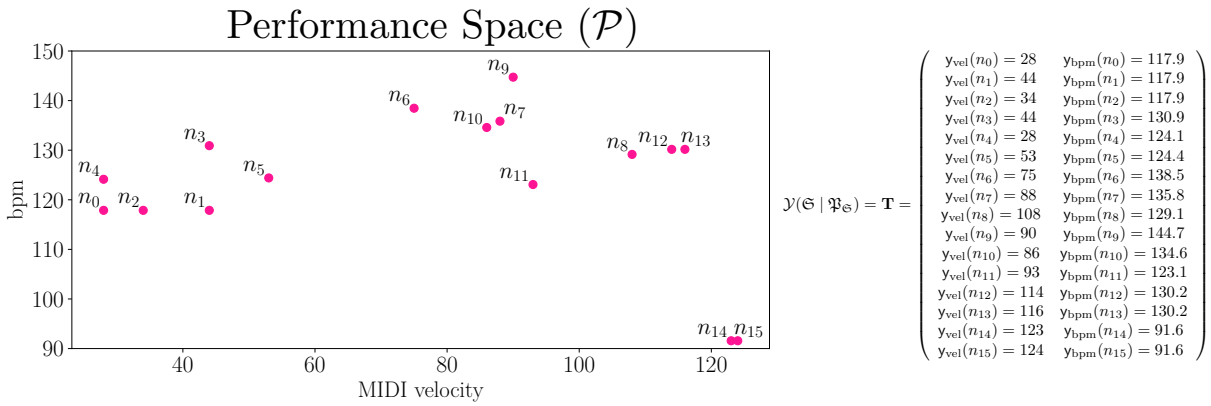


Figure 3.3: Performance space $\mathcal{P}$ spanned by performance representation model $\mathcal{Y} = \{\mathsf{y}_{\text{vel}}, \mathsf{y}_{\text{bpm}}\}$ for the matched performance shown in Figure 3.2. Each point in the performance space correspond to the representation of the performance of a score note n, i.e. a row in performance representation $\mathbf{T}$.

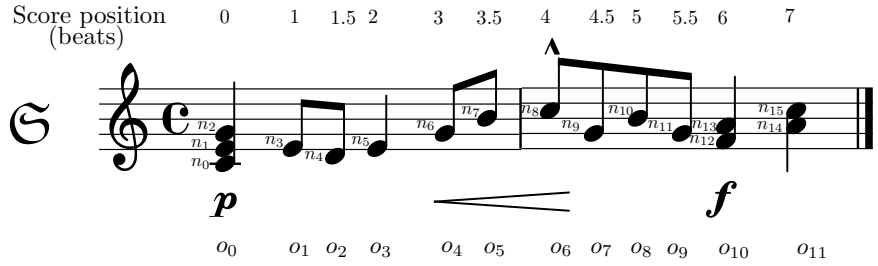
component is required: a method that, given a performance representation and its corresponding score, reconstructs an expressive performance in the same format as that which the performance representation model operates in (for example in MIDI format). This idea can be formalized as follows:

Definition 3.3 (Performance Decoder, Performance Codec, Lossless Representation)**.**

1. A *performance decoder* is a function $\mathcal{Y}^{-1}\colon (\mathcal{P}, \mathfrak{S}) \mapsto \mathbb{P}_{\mathfrak{S}}$ that given $\mathcal{Y}(\mathfrak{S} \mid \mathfrak{P}_{\mathfrak{S}}) = \mathbf{T} \in \mathbb{P}$, a performance representation under representation model $\mathcal{Y}$ and its corresponding score $\mathfrak{S}$, produces a matched performance of the same score $\mathcal{Y}^{-1}(\mathbf{T} \mid \mathfrak{S}) = \mathfrak{P}^*_{\mathfrak{S}}$, with $\mathfrak{P}^*_{\mathfrak{S}} \in \mathbb{P}_{\mathfrak{S}}$. In other words, a performance decoder takes the encoding of an expressive performance $\mathfrak{P}_{\mathfrak{S}}$ in terms of the expressive parameters defined in model $\mathcal{Y}$ (and its corresponding score) to reconstruct the performance in its original format. Note that $\mathfrak{P}_{\mathfrak{S}}$ and $\mathfrak{P}^*_{\mathfrak{S}}$ are not necessarily identical, i.e. the reconstructed performance is not guaranteed to be perfect (see point 3 below).
2. A *performance codec* is a tuple $\mathcal{C} = (\mathcal{Y}, \mathcal{Y}^{-1})$ consisting of a performance representation model $\mathcal{Y}$ and a performance decoder $\mathcal{Y}^{-1}$.
3. A performance codec $\mathcal{C}$ is said to be *lossless* if its performance representation model $\mathcal{Y}$ and its performance decoder allow for a perfect reconstruction of a performance i.e. $\mathcal{Y}^{-1}(\mathcal{Y}(\mathfrak{S} \mid \mathfrak{P}_{\mathfrak{S}}) \mid \mathfrak{S}) = \mathfrak{P}_{\mathfrak{S}}$, the case where the performance $\mathfrak{P}^*_{\mathfrak{S}}$ generated by the decoder is identical to performance $\mathfrak{P}_{\mathfrak{S}}$.

The performance codec in the example above is not lossless: using only MIDI velocity and bpm is insufficient to fully reconstruct the original MIDI file. The local tempo information represented by the bpm is not enough to compute the exact performed onset times and durations of each note. A performance codec that allows for fully reconstructing a performance would normally be desirable when modeling musical expression, because it provides a baseline against which model predictions can be accurately and precisely evaluated.

Note that a lossless performance codec does not guarantee that the expressive parameters chosen to represent the performance are musically meaningful, nor does it make constructing models that lead to good predictions easier. For example, the absolute performed onset time of each note is a parameter that can be directly used to reconstruct a performance, but is less meaningful than predicting expressive tempo, and is much more complicated to accurately predict.



Note Elements

Note	pitch(·)	onset(·)	duration(·)
n_0	60	0	1
n_1	64	0	1
⋮	⋮	⋮	⋮
n_{15}	72	7	1

Onset	Notes	onset(·)	IOI^{score}
o_0	$\{n_0, n_1, n_2\}$	0	1
o_1	$\{n_3\}$	1	0.5
⋮	⋮	⋮	⋮
o_{11}	$\{n_{14}, n_{15}\}$	7	1

Non-note Elements

Element	Type	start(·)	end(·)	Effect on
𝐂	time signature	0	8	$\{o_1, \ldots, o_{11}\}$
C major	key signature	0	8	$\{o_1, \ldots, o_{11}\}$
p	dynamics marking	0	6	$\{o_1, \ldots, o_9\}$
f	dynamics marking	6	8	$\{o_{10}, o_{11}\}$
<	dynamics marking	3	4.5	$\{o_4, \ldots, o_6\}$
∧	accent	4	4	$\{n_8\}$
⋮	⋮	⋮	⋮	⋮

Figure 3.4: Elements of Score $\mathfrak{S}$. Score notes are $\boldsymbol{\mathcal{X}} = \{n_1, \ldots, n_{15}\}$ and score onsets $\boldsymbol{\mathcal{O}} = \{o_1, \ldots, o_{11}\}$.

To unclutter notation, in the rest of this discussion we will write $\mathsf{y}(x_i \mid \mathfrak{P}_{\mathfrak{S}})$ as $\mathsf{y}(x_i)$.

3.2.2 Score Features: Score Basis Functions

Capturing the information contained in a musical score (i.e. the score features) and representing it numerically might be even less straightforward than representing performance information. After all, a score consists of symbols that hold both syntactic and semantic information, which has to be interpreted by the performer. Indeed, a good chunk of research in computational models of expressive performance, as well as music information retrieval (MIR) and computational musicology is dedicated to finding adequate representations to capture score information[8].

Figure 3.4 presents an example of the different elements of a score $\mathfrak{S}$. From this figure we can see that while certain elements of the score might have a straightforward numerical representation (e.g. certain properties of note elements such as notated duration and MIDI pitch[9]), other

[8] For example, see Conklin and Witten (1995), van Kranenburg and Backer (2004) and Velarde et al. (2018) for three very different approaches to representing score information. For a more in-depth discussion of representing score features in computational models of expressive performance see Section 9.3.1 and 9.3.3 and references therein.

[9] Although MIDI pitch might be the most widespread representation of the chromatic pitch of a note it is not without its faults. Readers familiar with music theory could point out that by representing enharmonically equivalent notes with the same MIDI pitch (e.g. both D♭4 and C♯4 are represented as MIDI pitch 61), important

elements like metrical information and dynamics do not.

In order to extract and combine information from different elements in the score, we borrow the concept of *basis functions* from the machine learning literature (Bishop, 2006). For example, we can define a function pitch(·) that maps notes in score $\mathfrak{S}$ to their MIDI pitch, and a function forte(·) that maps notes in the score to 1 if they belong to the onsets for which $\boldsymbol{f}$ has an effect (in the example in Figure 3.4, all notes that belong to onsets o_{10} and o_{11}), and zero otherwise. We can formalize this idea as follows:

Definition 3.4 (Score Basis Functions)**.** A *score basis function* is a function $\varphi\colon \mathfrak{S} \mapsto \mathbb{R}$ that captures a structural aspect of the score and expresses the relation of each element in the score to that aspect. The value of the score basis function for element $x_i \in \mathfrak{S}$ is denoted as $\varphi(x_i)$.

Continuing with the example above, $\varphi_{\text{pitch}}(n_i) = \text{pitch}(n_i)$ and $\varphi_{\boldsymbol{f}}(n_i) = \text{forte}(n_i)$ would be score basis functions encoding pitch information of each note, and information about the dynamics marking $\boldsymbol{f}$ for score note n_i, respectively. It is important to note that the definition of score basis functions, in particular for those functions encoding mid- and high-level properties of the score, is a theory-laden activity, since it relies on our knowledge and/or interpretation of the meaning of the score feature that we want to represent. An important part of this thesis is dedicated to describing different groups of score basis functions, in particular Sections 3.3.2, 3.4.1, 4.2.1, 4.3.3. Chapters 5 and 6 are dedicated to studying basis functions for describing orchestral music and cognitively motivated features, respectively. Appendix A.1 contains a list of all basis functions defined in this work. A related representation to that of the score basis functions is the *multiple viewpoint* system (Conklin and Witten, 1995; Conklin, 2013).

Using these score basis functions, we can construct a space similar to the performance space for numerically representing the information contained in the score as follows:

Definition 3.5 (Score Space, Score Representation)**.**

1. A *score representation model* is a set of score basis functions, which we denote as $\mathcal{F} = \{\varphi_1, \ldots, \varphi_M\}$. In a similar fashion to the performance representation model, we will slightly abuse notation and write $\mathcal{F}(x_i) = (\varphi_1(x_i), \ldots, \varphi_M(v_i))^{\mathrm{T}} = \boldsymbol{\varphi}(x_i)$, with $\boldsymbol{\varphi}(x_i) \in \mathbb{R}^M$, as a vector valued function $\mathcal{F}\colon \mathfrak{S} \mapsto \mathcal{S}$, which maps element x_i of the score $\mathfrak{S}$ into the *score space*, a vector subspace $\mathcal{S} \subset \mathbb{R}^M$ whose dimensions represent the numerical encodings of structural aspects of the score captured by the expressive basis functions. In other words, $\boldsymbol{\varphi}(x_i)$ is a vector whose j-th component is given by $\varphi_j(x_i)$, the j-th score basis function evaluated for score element x_i. For the sake of clarity, whenever we are referring to a specific score basis function we will write $\varphi_{\text{name of the function}}$ instead of simply writing $\varphi_{\textit{parameter index}}$, as done in the example above with φ_{pitch} and $\varphi_{\boldsymbol{f}}$ instead of φ_1 and φ_2.

2. The *score representation of* $\mathfrak{S}$ *under* $\mathcal{F}$ (or simply *score representation of* $\mathfrak{S}$), denoted as $\mathcal{F}(\mathfrak{S}) = \boldsymbol{\Phi}$, is the value of the basis functions for all relevant elements of $\mathfrak{S}$. In a similar fashion to the one discussed for performance representations, in most cases $\boldsymbol{\Phi}$ is a real-valued matrix in $\mathbb{R}^{N \times M}$, where its i-th row is given by $\boldsymbol{\varphi}(x_i)$, the value of all score basis functions for score element x_i. In Chapter 4, we will explore cases where some score basis functions are defined at the note level (e.g. like the pitch of a note) while others are defined at the onset level (e.g. the position within the bar of a given score position), where $\boldsymbol{\Phi} = \{\boldsymbol{\Phi}_{\text{onset-wise}}, \boldsymbol{\Phi}_{\text{note-wise}}\}$ is a composite structure, with $\boldsymbol{\Phi}_{\text{onset-wise}} \in \mathbb{R}^{N_{No} \times M_{\text{ow}}}$ and $\boldsymbol{\Phi}_{\text{note-wise}} \in \mathbb{R}^{N_{Nx} \times M_{\text{nw}}}$, where M_{ow} and M_{nw} are the number of score basis functions

information regarding the tonal function of the notes gets lost.

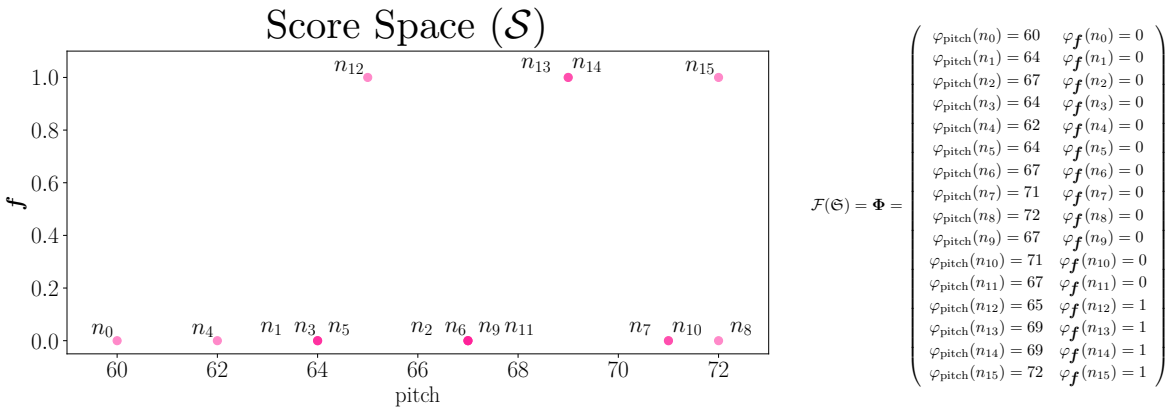


Figure 3.5: Score space $\mathcal{S}$ spanned by score representation model $\mathcal{F} = \{\varphi_{\text{pitch}}, \varphi_{\boldsymbol{f}}\}$ for the score shown in Figure 3.4. Each point in the space correspond to the representation of a score note n, i.e. a row of $\mathbf{\Phi}$. Note that due to the chosen representation model, the points for some of the notes overlap.

defined onset-wise and note-wise, respectively.

Continuing with the example, Figure 3.5 shows the score space $\mathcal{S}$ for score representation model $\mathcal{F} = \{\varphi_{\text{pitch}}, \varphi_{\boldsymbol{f}}\}$. In this plot, each point corresponds to $\mathcal{F}(n_i)$, i.e., the representation of score note n_i.

In the rest of this thesis, we will use *basis functions* as a shorthand for score basis functions.

3.2.3 Computational Model

Sections 3.2.1 and 3.2.2 describe a formalization of the concepts of expressive parameters and score features described in Section 2.3. The last piece of the puzzle are the computational models.

As described in Section 2.3.3, a computational model is a function that, given features representing score information, *makes predictions* of the value of the expressive parameters. Continuing with our running example, we can use a simple linear model (for instance) to relate the score representation $\mathbf{\Phi}$ shown in Figure 3.5 to the performance representation $\mathbf{T}$ shown in Figure 3.3 as follows: for expressive dynamics, represented by MIDI velocity, we define a linear function as

$$\begin{aligned} f^{(\text{vel})}(\boldsymbol{\varphi}(n_i)) &= w^{(\text{vel})}_{\text{pitch}} \varphi_{\text{pitch}}(n_i) + w^{(\text{vel})}_{\boldsymbol{f}} \varphi_{\boldsymbol{f}}(n_i) + w^{(\text{vel})}_0 \\ &= 5.8 \cdot \varphi_{\text{pitch}}(n_i) + 41 \cdot \varphi_{\boldsymbol{f}}(n_i) - 318.7; \end{aligned}$$

and for expressive tempo, represented by the local bpm, we define a similar linear function as

$$\begin{aligned} f^{(\text{bpm})}(\boldsymbol{\varphi}(n_i)) &= w^{(\text{bpm})}_{\text{pitch}} \varphi_{\text{pitch}}(n_i) + w^{(\text{bpm})}_{\boldsymbol{f}} \varphi_{\boldsymbol{f}}(n_i) + w^{(\text{bpm})}_0 \\ &= 0.3 \cdot \varphi_{\text{pitch}}(n_i) - 18.1 \cdot \varphi_{\boldsymbol{f}}(n_i) + 108. \end{aligned}$$

We refer to $\boldsymbol{\theta}_{\text{vel}} = \left\{w^{(\text{vel})}_{\text{pitch}}, w^{(\text{vel})}_{\boldsymbol{f}}, w^{(\text{vel})}_0\right\}$ and $\boldsymbol{\theta}_{\text{bpm}} = \left\{w^{(\text{bpm})}_{\text{pitch}}, w^{(\text{bpm})}_{\boldsymbol{f}}, w^{(\text{bpm})}_0\right\}$ as the parameters of $f^{(\text{vel})}$ and $f^{(\text{bpm})}$, respectively[10]. We can then define a vector function $F(\varphi(n_i)) =$

[10] The values of the parameters in this example were estimated by fitting the performance using the least squares algorithm, which will be discussed in detail in Section 3.3.3 below.

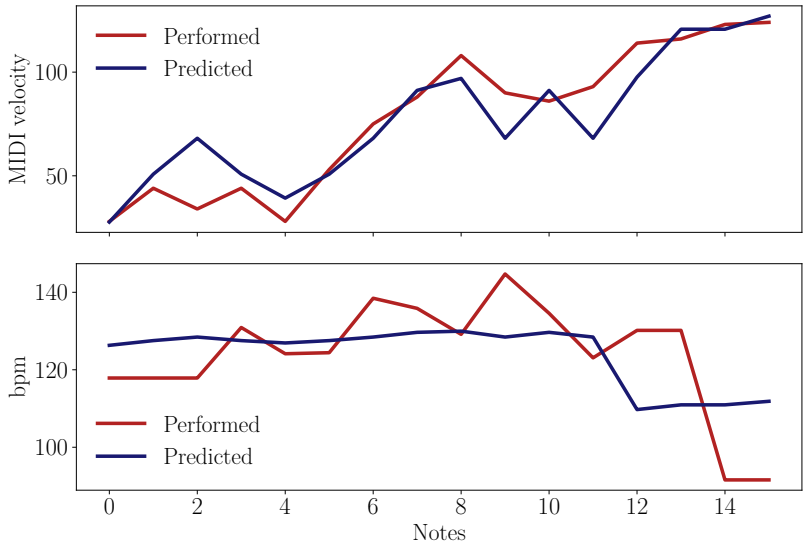


Figure 3.6: Comparison between predicted ($\mathbf{Y}$, in blue) and performed ($\mathbf{T}$, in red) expressive parameters for the performance representation model $\mathcal{Y} = \{\mathsf{y}_{\text{vel}}, \mathsf{y}_{\text{bpm}}\}$ described in Figure 3.3. The predictions were computed using a linear model with the score representation $\mathcal{F} = \{\varphi_{\text{pitch}}, \varphi_{\boldsymbol{f}}\ \}$ described in Figure 3.5.

$\left(f^{(\text{vel})}(\boldsymbol{\varphi}(n_i)), f^{(\text{bpm})}(\boldsymbol{\varphi}(n_i))\right)^{\mathrm{T}}$, whose output, denoted as $\mathbf{Y}$, predicts the values of the performance representation $\mathbf{T}$. Figure 3.6 shows a comparison between performance representation $\mathbf{T}$ and the predictions of the model $\mathbf{Y}$.

Using this notion of mathematical functions taking score information encoded numerically (using basis functions) as input for predicting the value of the expressive parameters, we can finally formalize a basis function model as follows:

Definition 3.6 (Basis Function Model)**.** A Basis Function model is a triple $\text{BM} = (\mathcal{C}, \mathcal{F}, F(\cdot\,; \boldsymbol{\theta}))$ consisting of:

1. $\mathcal{C} = (\mathcal{Y}, \mathcal{Y}^{-1})$, a performance codec consisting of
   - $\mathcal{Y}\colon \mathfrak{P}_{\mathfrak{S}} \mapsto \mathcal{P}$, a performance representation model, and
   - $\mathcal{Y}^{-1}$, its corresponding decoder;
2. $\mathcal{F}\colon \mathfrak{S} \mapsto \mathcal{S}$, a score representation model; and
3. $F(\cdot\,; \boldsymbol{\theta})\colon \mathcal{S} \mapsto \mathcal{P}$, a function with parameters $\boldsymbol{\Theta}$, which we call *predictive function*, that maps points in the score space $\mathcal{S}$ to points in the performance space $\mathcal{P}$. In other words, F maps a score $\mathfrak{S}$ represented in terms of basis functions to a performance represented in terms of expressive parameters (which can then be formed into an expressive performance $\mathfrak{P}_{\mathfrak{S}}$ via the decoder $\mathcal{Y}^{-1}$).

   In cases where all basis functions and expressive parameters act on the same level (onset-wise or note-wise), we can write this function as

$$F(\boldsymbol{\varphi}(x_i)\,; \boldsymbol{\Theta}) = \left(f^{(1)}(\boldsymbol{\varphi}^{(1)}(x_i), \boldsymbol{\theta}_1), \ldots, f^{(N_y)}(\boldsymbol{\varphi}^{(N_y)}(x_i)\,; \boldsymbol{\theta}_{N_y})\right)^{\mathrm{T}},$$

   where

- $\boldsymbol{\varphi}^{(j)}(x_i)$ is a vector whose components are the values of the subset of all basis functions in the score representation $\mathcal{F}$ that are relevant for predicting the j-th expressive parameter for score element x_i (and $\boldsymbol{\varphi}(x_i)$ contains the value of all basis functions in score representation $\mathcal{F}$ for score element x_i).

- Each function $f^{(j)}(\cdot\,;\boldsymbol{\theta}_j)$ with parameters $\boldsymbol{\theta}_j$ predicts the values of the j-th expressive parameter, i.e. maps points in the score space $\mathcal{S}$ to the j-th coordinate of the performance space $\mathcal{P}$. The output of these predictive functions is denoted as $f^{(j)}(\boldsymbol{\varphi}^{(j)}(x_i)\,;\boldsymbol{\theta}_j) = y_i^{(j)}$. For the sake of clarity, whenever we are referring to a specific expressive parameter we will write $f^{(\text{name of the parameter})}$ instead of simply writing $f^{(\textit{parameter index})}$ (e.g. in the example above we wrote $f^{(\text{vel})}$ and $f^{(\text{bpm})}$ instead of $f^{(1)}$ and $f^{(2)}$).

- The full set of parameters is $\boldsymbol{\Theta} = \bigcup_{j=1}^{N_y} \boldsymbol{\theta}_j$.

We will abuse notation and write $F(\boldsymbol{\Phi}\,;\boldsymbol{\Theta}) = \mathbf{Y}$, where $\mathbf{Y}$ is an object with the same structure as $\mathbf{T}$ (i.e., in most cases a real valued matrix in $\mathbb{R}^{N\times N_y}$) that represents the estimated performance of score $\mathfrak{S}$. In the rest of this discussion, we refer to $\mathbf{Y}$ as the *predicted performance representation under* $\mathcal{Y}$ (or simply *predicted performance*).

4. A BM models an expressive performance as a *regression problem*: We assume that the value of the j-th expressive parameter for score element x_i can be written as

$$\mathsf{y}_j(x_i) = f^{(j)}(\boldsymbol{\varphi}^{(j)}(x_i)\,;\boldsymbol{\theta}_j) + \epsilon \tag{3.1}$$

where ϵ is a random variable representing an error term. A common choice for ϵ is a zero mean Gaussian random variable drawn from $\mathcal{N}(0,\sigma^2)$, where σ^2 is the variance of the distribution.

Note that Definition 3.6 allows predictive functions for each expressive parameter to have their own set of input basis functions. This property is motivated by the fact that, in practice, not all score features might be relevant to predict all expressive parameters (e.g. pitch related features might be more relevant to predict dynamics, whereas metrical related features are more useful for predicting expressive tempo). However, in the following discussion, for the sake of simplicity and unless otherwise specified, we will assume that $\boldsymbol{\varphi}^{(1)}(x_i) = \cdots = \boldsymbol{\varphi}^{(N_y)}(x_i) = \boldsymbol{\varphi}(x_i)$.

Although the general definition of BMs allows for joint modeling of expressive parameters, in practice it is easier to construct and evaluate independent models for each parameter. Section 9.4.1 presents a brief overview of the state of the art in joint modeling of expressive parameters.

Assuming we get a set of N_p pieces for training the model $\boldsymbol{\mathcal{T}} = \{(\mathbf{T}_1,\boldsymbol{\Phi}_1),\ldots,(\mathbf{T}_{N_p},\boldsymbol{\Phi}_{N_p})\}$, where each element is a tuple consisting of performance and score representations under the same performance representation model $\mathcal{Y}$ and score representation model $\mathcal{F}$, we can use mathematical optimization techniques to learn the parameters that minimize a loss function $\mathcal{L}$, i.e.

$$\hat{\boldsymbol{\Theta}} = \underset{\boldsymbol{\Theta}}{\operatorname{argmin}}\, \mathcal{L}(\boldsymbol{\mathcal{T}},\boldsymbol{\Theta}). \tag{3.2}$$

A common choice of loss function for regression problems is the squared error. In this work, we will discuss specific choices of performance and score representation models, as well as predictive/modeling functions and loss functions.

3.2.4 A Note on Evaluation Strategies for Computational Models of Musical Expression

It is important to emphasize that the squared error (or other derived measures) with respect to a human performance is not necessarily a reliable indicator of the musical quality of a generated performance. Firstly, a low error does not always imply that a performance *sounds* good—at crucial places, a few small errors may make the performance sound severely awkward, even if these error hardly affect the overall error measure. Secondly, the very notion of expressive freedom implies that music can often be performed in a variety of ways that are very different, but equally convincing in their own right. In that sense, evaluating the models only by their ability to predict the performance of a single performer is not sufficient in the long run.

In spite of that, there are good reasons in favor of the mean squared error as a guide for evaluating performance model. Firstly, music performance is not *only* about individual expressive freedom.

The works of Repp (e.g. 1992; 1994) have shown that there are substantial commonalities in the variations of timing and dynamics across performers. We believe that assessing how well the model predicts existing human performances numerically does tell us something about the degree to which it has captured general performance principles.

Secondly, model assessment involving human judgments of the perceptual validity of output is a time-consuming and costly effort, not practicable as a recurring part of an iterative model development/evaluation process. For such a process, numerical benchmarking against performance corpora is more appropriate. Note however that anecdotal perceptual validation of the BMs has taken place on several occasions, in the form of concerts/competitions where the BM, along with competing models was used to perform musical pieces live in front of an audience, on a computer-controlled grand piano[11].

3.3 Linear Basis Models

As previously stated, we start the discussion on modeling choices chronologically. The original formulation of the BM framework, the *linear basis models* (LBM) was proposed by Grachten and Widmer (2012). LBMs are models that map score features (represented by basis functions) into the expressive parameters as a linear combination of the values of the basis functions. LBMs are note-wise models, i.e. they make predictions of the expressive parameters for each note in the score. LBMs were designed for modeling piano music of the common practice period, and thus, some design choices, in particular choices for expressive parameters and basis functions are made specifically for this purpose.

The rest of this section is structured as follows: In Section 3.3.1 we discuss the encoding of the expressive parameters. Section 3.3.2 presents a brief overview of the groups of basis functions used in the first version of the LBM framework. Finally Section 3.3.3 provides a mathematical characterization of the LBMs.

[11] Predecessors of the current BM approach have won awards at the 2008 and 2011 editions of the Music Performance Rendering Contest (RenCon) (Katayose et al., 2012), and was evaluated favorably in a Turing test-like concert organized as part of the 1st International Workshop on Computer and Robotic Systems for Automatic Music Performance (SAMP14) (Rodà et al., 2015), as well as in a musical Turing test reported in (Schubert et al., 2017).

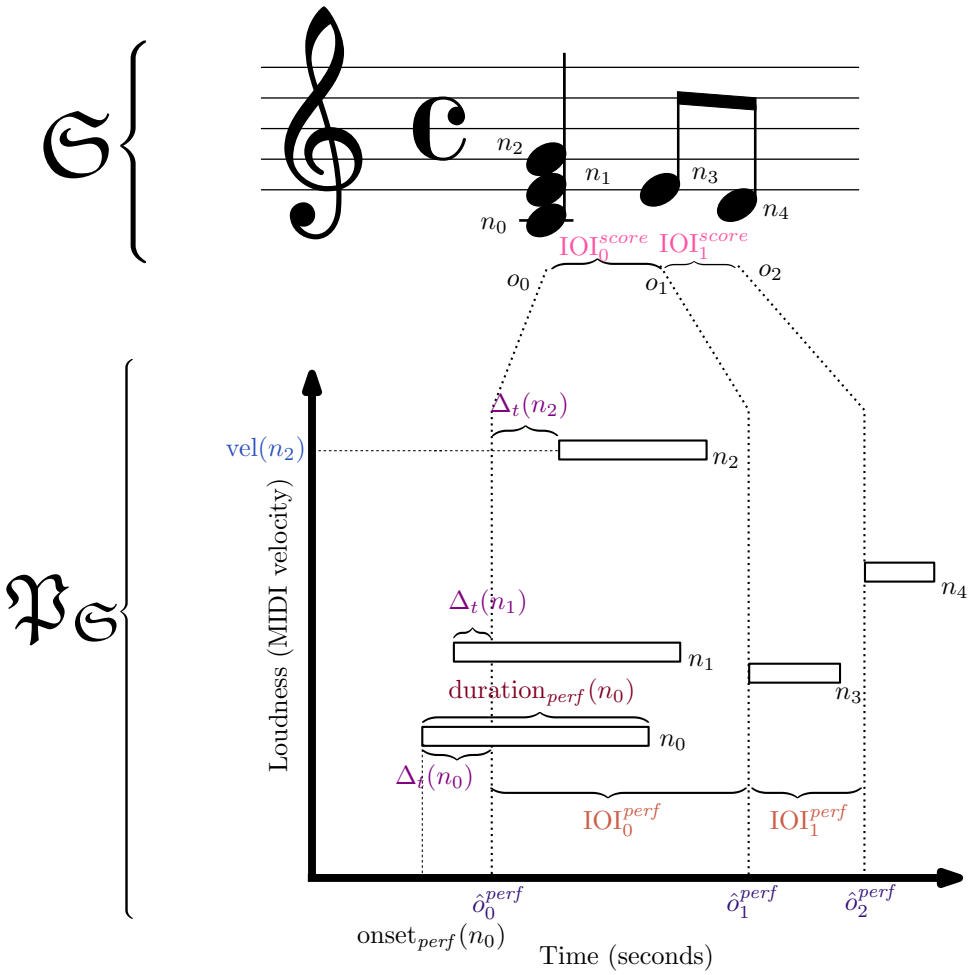


Figure 3.7: Excerpt of a matched MIDI performance $\mathfrak{P}_{\mathfrak{S}}$ (as a piano-roll where the x axis describes performance time and the y axis describes the MIDI velocity) and its corresponding score $\mathfrak{S}$ showcasing elements for computing the expressive parameters described in performance codec v. 1.0.

3.3.1 Expressive Parameters: Performance Codec v. 1.0

As mentioned in Section 2.3.2, defining the parameters that describe an expressive performance is a *theory-laden* activity, since the choice of these parameters is intrinsically dependent on the assumptions made by the researchers about the performance as a process. In this section we discuss the first version of the performance codec, which we denote as $\mathcal{C}_{1.0} = (\mathcal{Y}_{1.0}, \mathcal{Y}_{1.0}^{-1})$, where $\mathcal{Y}_{1.0}$ is the performance representation model and $\mathcal{Y}_{1.0}^{-1}$ the performance decoder, which are described below. As previously stated, LBMs are note-wise models, and thus, they make predictions of the expressive parameters for each note in the score. Figure 3.7 presents a visual representation of an excerpt of a matched performance in MIDI format which highlights the elements for computing the expressive parameters described below. The definition of the following parameters follows the notation conventions established in Section 3.2 and illustrated in Figures 3.2 and 3.4.

Performance Representation Model

The first version of the performance representation model v. 1.0 (denoted as $\mathcal{Y}_{1.0}$) consists of four expressive parameters, which are defined as follows:

1. **MIDI velocity.** Dynamics relates to the changes in loudness of the performance. In the case of piano performances, the loudness is a function of the hammer-velocities of the notes as they are performed by the pianist. Of course, the relation between loudness, which is a subjective measure, and the hammer-velocities, represented as MIDI values between 0 and 127, may not be linear (see (Goebl and Bresin, 2003) for a discussion of the mechanical aspects involved in this relation). For simplicity, however, we treat the MIDI velocity of the performed notes as a proxy for dynamics, so that the values can be extracted per note directly from the corpus. In this way, we can estimate the loudness of a performed note as

$$\mathrm{y}_{\mathrm{vel}}(n_i) = \frac{\mathrm{vel}(n_i)}{127}, \tag{3.3}$$

where $\mathrm{vel}(n_i)$ is the performed MIDI velocity of score note n_i (e.g. see $\mathrm{vel}(n_2)$ in blue in Figure 3.7).

2. **log BPR.** Musical tempo is defined as the rate at which musical beats occur. Fluctuations in the beat rate over short time-spans (corresponding roughly to the time-spans of musical groupings such as phrases and sub-phrases) are referred to as changes in *local tempo*. Note that local tempo is a *virtual* quantity that can not be measured directly, but is implied by the temporal position of note events in the performance.

   Rather than the beat rate, we take its reciprocal, the *beat period* (BP), also referred to as *inter-beat interval* (IBI), as a representation of local tempo, which can be computed as

$$\mathrm{BP}(n_i) = \frac{\mathrm{IOI}^{perf}_{o(n_i)}}{\mathrm{IOI}^{score}_{o(n_i)}}, \tag{3.4}$$

   where $\mathrm{IOI}^{perf}_{o(n_i)}$ and $\mathrm{IOI}^{score}_{o(n_i)}$ are the performed and notated inter-onset intervals (IOIs) corresponding to the score onset $o(n_i)$ to which score note n_i belongs, respectively (see column "Notes" in the table in the right in Figure 3.4). In this formulation we choose to assign the value of the IOI between score positions o_k and o_{k+1} to o_k, i.e. the "beginning" of the IOI, as showed in Figure 3.4 in column IOI^{score} and illustrated in Figure 3.7, with IOI^{score}_0 and IOI^{score}_1 (in pink) representing the score IOIs, and IOI^{perf}_0 and IOI^{perf}_1 (in red) representing the performed IOIs corresponding to score onsets o_0 and o_1, respectively.

   We compute the notated IOI between score onsets o_k and o_{k+1} as

$$\mathrm{IOI}^{score}_{o_k} = \mathrm{onset}(o_{k+1}) - \mathrm{onset}(o_k). \tag{3.5}$$

   The performed IOI corresponding to the above defined $\mathrm{IOI}^{score}_{o_k}$ is computed as

$$\mathrm{IOI}^{perf}_{o_k} = \hat{o}^{perf}_{k+1} - \hat{o}^{perf}_{k} \tag{3.6}$$

   where $\hat{o}^{perf}_k$ is the *equivalent performed onset time* (or simply *equivalent onset*) in seconds corresponding to score onset o_i. For the first version of the expressive parameters, this equivalent performed onset time was simply defined as the average onset of all notes belonging to score onset o_k, i.e.,

$$\hat{o}^{perf}_k = \frac{1}{|o_k|} \sum_{n_l \in o_k} \mathrm{onset}_{perf}(n_l). \tag{3.7}$$

   This formulation of the equivalent performed onset is illustrated for $\hat{o}^{perf}_0$, $\hat{o}^{perf}_1$ and $\hat{o}^{perf}_2$ in indigo in Figure 3.7. Since we define the BP in a note-wise fashion, all notes in score

onset o_k have the same BP (e.g. $\text{BP}(n_0) = \text{BP}(n_1) = \text{BP}(n_2)$ in the example in Figure 3.7).

We are interested in the relative local tempo values within a piece, rather than absolute tempo values. For this reason, we normalize the computed BP values by dividing them by the average BP over the performance of a piece (BP_{ave}). We refer to this normalized BP as the *beat period ratio* (BPR). This results in a representation of local tempo where a BPR value of 1 corresponds to the average tempo, a BPR value of $\frac{1}{2}$ to double the average tempo, and a BPR value of 2 to half the average tempo. To make equal increments in BPR values correspond to the same proportional decrease in tempo, irrespective of the average tempo, we take the 2-logarithm of the BPR values, such that an increment of 1 always corresponds to a decrease in tempo by a factor of 2, i.e.,

$$\mathsf{y}_{\log \text{bpr}}(n_i) = \log_2 \frac{\text{BP}(n_i)}{\text{BP}_{ave}}. \tag{3.8}$$

It is important to mention that there are several factors that might complicate the computation of meaningful local tempo values from the note events in the performance as described above. First, the computation of BP described above can lead to erratic jumps of subsequent BP values, especially when the score IOIs are relatively small. Second, there is evidence that perceived beat times do not always coincide with the onset time of a note event at that beat: listeners prefer a beat grid that is slightly smoother than implied by the literal note onsets (Dixon et al., 2006). The search for a more cognitively plausible representation of musical tempo is one of the main motivations behind the definition of the expressive parameters described in Section 6.3.1 and Appendix B. For a statistical perspective on the problem of finding equivalent onsets see (Fu et al., 2015).

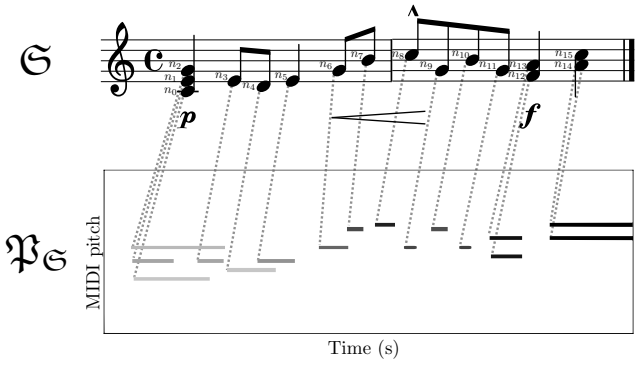
3. **Timing.** The above definition of local tempo implies that note onset times in a performance can be shifted with respect to the implied beat grid. We refer to this temporal shifting of events as timing (following (Honing, 2001)). Timing encompasses different expressive phenomena, such as *chord spread* (Fu et al., 2015), and *melody lead* (Goebl, 2001).

   The timing of a note n_i can be computed by first estimating the equivalent onset of that event, calculated as average onset time of all notes in that score onset, denoted as $\hat{o}^{perf}(n_i)$, and taking the difference between that onset and the actual onset in the performance, denoted as $\text{onset}_{perf}(n_i)$, such that the anticipation of an event leads to a positive timing value, and the delay of an event leads to a negative timing value, i.e.,

$$\begin{aligned} \mathsf{y}_{\text{tim}}(n_i) &= \Delta_t(n_i) \\ &= \hat{o}^{perf}(n_i) - \text{onset}_{perf}(n_i) \end{aligned} \tag{3.9}$$

   Figure 3.7 illustrates the above idea of temporal shifting for notes n_0, n_1 and n_2 ($\Delta_t(n_0)$, $\Delta_t(n_1)$ and $\Delta_t(n_2)$ in violet). Note that the above definition of BPR implies that the timing for score onsets consisting of a single note (as is the case of o_1 and o_2 in our running example) is exactly zero, as illustrated in Figure 3.7 for n_3 and n_4 ($\hat{o}_1^{perf}$ and $\hat{o}_2^{perf}$ in indigo).

4. **log-Articulation.** Articulation refers to the lengthening or shortening of note durations in a performance relative to the notated duration and the current local tempo, producing playing styles such as *legato* and *staccato*. We compute the articulation of a note by dividing the actual duration of a performed note $\text{duration}_{perf}(n_i)$ (illustrated in Figure 3.7 in burgundy for n_0) by its reference duration $\text{duration}(n_i)$. As with the computation of



$\mathcal{Y}_{1.0}(\mathfrak{S} \mid \mathfrak{P}_{\mathfrak{S}}) = \mathbf{T}$

Note	$\mathsf{y}_{\text{vel}}(\cdot)$	$\mathsf{y}_{\log \text{bpr}}(\cdot)$	$\mathsf{y}_{\text{tim}}(\cdot)$	$\mathsf{y}_{\log \text{art}}(\cdot)$
n_0	0.220	0.091	-0.014	0.230
n_1	0.346	0.091	0.000	-0.624
n_2	0.268	0.091	0.014	0.551
n_3	0.346	-0.061	0.000	-0.152
n_4	0.220	0.016	0.000	0.697
n_5	0.417	0.013	0.000	-0.690
n_6	0.591	-0.142	0.000	0.113
n_7	0.693	-0.114	0.000	-0.619
n_8	0.850	-0.041	0.000	-0.534
n_9	0.709	-0.206	0.000	-1.037
n_{10}	0.677	-0.101	0.000	-0.675
n_{11}	0.732	0.028	0.000	-1.379
n_{12}	0.898	-0.053	-0.002	-0.889
n_{13}	0.913	-0.053	0.002	-0.841
n_{14}	0.969	0.455	-0.002	0.023
n_{15}	0.976	0.455	0.002	0.000

Figure 3.8: Performance representation $\mathbf{T}$ under representation model $\mathcal{Y}_{1.0}$ for matched performance $\mathfrak{P}_{\mathfrak{S}}$ (with score $\mathfrak{S}$) introduced in the running example in Section 3.2 (see Figure 3.2).

log-BPR values, we take the base 2 logarithm of this ratio, such that a doubling of the duration always corresponds to an articulation increase of 1. This parameter is computed as

$$\mathsf{y}_{\log \text{art}}(n_i) = \log_2 \frac{\text{duration}_{perf}(n_i)}{\text{duration}(n_i)\text{BP}(n_i)}. \tag{3.10}$$

The performance representation of $\mathfrak{P}_{\mathfrak{S}}$ under the representation model $\mathcal{Y}_{1.0} = \{\mathsf{y}_{\text{vel}}, \mathsf{y}_{\log \text{bpr}}, \mathsf{y}_{\text{tim}}, \mathsf{y}_{\log \text{art}}\}$ defined by the above is an $N_x \times 4$ matrix given as

$$\mathcal{Y}_{1.0}(\mathfrak{S} \mid \mathfrak{P}_{\mathfrak{S}}) = \mathbf{T} = \begin{pmatrix} \mathsf{y}_{\text{vel}}(n_0) & \mathsf{y}_{\log \text{bpr}}(n_0) & \mathsf{y}_{\text{tim}}(n_0) & \mathsf{y}_{\log \text{art}}(n_0) \\ \vdots & \vdots & \vdots & \vdots \\ \mathsf{y}_{\text{vel}}(n_{N_x-1}) & \mathsf{y}_{\log \text{bpr}}(n_{N_x-1}) & \mathsf{y}_{\text{tim}}(n_{N_x-1}) & \mathsf{y}_{\log \text{art}}(n_{N_x-1}) \end{pmatrix}. \tag{3.11}$$

Figure 3.8 shows the full performance representation $\mathbf{T}$ under representation model $\mathcal{Y}_{1.0}$ defined above for the running example introduced in Section 3.2.

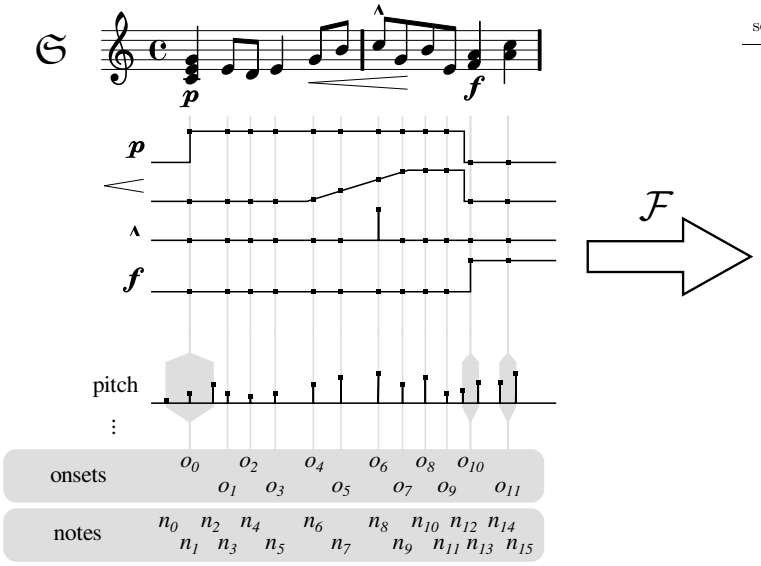
Performance Decoder

It is straightforward to show that representation model $\mathcal{Y}_{1.0}$ discussed above is lossless up to the average beat period BP_{ave}.

Given a performance representation $\mathbf{T}$, its score $\mathfrak{S}$, and the average beat period of the performance BP_{ave}, we can reconstruct a performance $\mathfrak{P}_{\mathfrak{S}}$ (in MIDI format) using performance decoder v. 1.0 (denoted as $\mathcal{Y}_{1.0}^{-1}$). This performance decoder estimates the measurable quantities describing the performance information of each note in the MIDI file discussed in Section 3.2.1, namely, MIDI pitch of each note, performed MIDI velocity, performed onset time in seconds and duration[12].

A detailed description of performance decoder $\mathcal{Y}_{1.0}^{-1}$, including the steps to compute the performance information in MIDI format given a performance representation, its score and the average

[12] As discussed in Section 3.2.1, we refer to the sounding duration of a note that includes pedal information, and not to time interval between `NOTE ON` and `NOTE OFF` messages.



$\mathcal{F}(\mathfrak{S}) = \mathbf{\Phi}$

score pos.	Note	φ_{pitch}	$\varphi_{\boldsymbol{p}}$	$\varphi_{\boldsymbol{f}}$	$\varphi_{\boldsymbol{cresc}}$	φ_{accent}
0.0	n_0	$\frac{60}{127}$	1.0	0.0	0.0	0.0
0.0	n_1	$\frac{64}{127}$	1.0	0.0	0.0	0.0
0.0	n_2	$\frac{67}{127}$	1.0	0.0	0.0	0.0
1.0	n_3	$\frac{64}{127}$	1.0	0.0	0.0	0.0
1.5	n_4	$\frac{62}{127}$	1.0	0.0	0.0	0.0
2.0	n_5	$\frac{64}{127}$	1.0	0.0	0.0	0.0
3.0	n_6	$\frac{67}{127}$	1.0	0.0	0.3	0.0
3.5	n_7	$\frac{71}{127}$	1.0	0.0	0.7	0.0
4.0	n_8	$\frac{72}{127}$	1.0	0.0	1.0	1.0
4.5	n_9	$\frac{67}{127}$	1.0	0.0	1.0	0.0
5.0	n_{10}	$\frac{71}{127}$	1.0	0.0	1.0	0.0
5.5	n_{11}	$\frac{67}{127}$	1.0	0.0	1.0	0.0
6.0	n_{12}	$\frac{65}{127}$	0.0	1.0	0.0	0.0
6.0	n_{13}	$\frac{69}{127}$	0.0	1.0	0.0	0.0
7.0	n_{14}	$\frac{69}{127}$	0.0	1.0	0.0	0.0
7.0	n_{15}	$\frac{72}{127}$	0.0	1.0	0.0	0.0

Figure 3.9: Left: Schematic view of basis functions representing score information for score $\mathfrak{S}$. Right: $\mathbf{\Phi}$, the score representation of $\mathfrak{S}$ under score representation model $\mathcal{F} = \{\varphi_{\boldsymbol{p}}, \varphi_{\boldsymbol{f}}, \varphi_{\boldsymbol{cresc}}, \varphi_{\text{accent}}\}$.

beat period of the performance is shown in Algorithm 3.1. A description of a method to write this performance information into an actual MIDI file lies beyond the scope of this discussion and is, thus, not included here.

3.3.2 Basis Functions

In the following, we describe various groups of basis functions, where each group represents a different aspect of the musical score. The information conveyed by the basis functions is either explicitly available in the score (such as dynamics markings, and pitch), or can be inferred from it in a straight-forward manner (such as metrical position and IOIs). This list should by no means be taken as an exhaustive (or accurate) set of features for modeling musical expression. Figure 3.9 shows a schematic view of some basis functions representing score information.

As previously mentioned, instead of simply describing the latest state of the BM framework, we describe its development in chronological order. Therefore, the following list of basis functions comprises features defined by Grachten and Widmer (2012), and thus, does not represent the complete list of basis functions. A formal description of all basis functions described in this work is available in Appendix A.1.

1. **Polynomial pitch model**. Grachten and Widmer (2012) proposed a third order polynomial model to describe the dependency of expressive parameters on pitch. This model can be integrated in the BM approach by defining each term in the polynomial as a separate basis function, i.e. "pitch", "pitch^2", and "pitch^3". This polynomial model originates from a statistical analysis of the dependence of the MIDI velocity on note pitch in the Magaloff/Chopin dataset. In Figure 3.10, a polynomial fitting of the notes for the Magaloff/Chopin dataset is shown.

2. **Dynamics markings**. Bases that encode dynamics markings, such as shown in Figure 3.9 for ***p***, ***f*** and ***crescendo***. These basis functions include performance directives that describe

Algorithm 3.1: Performance Decoder v. 1.0

Input:

- $\mathfrak{S}$: Score
  - $\boldsymbol{\mathcal{X}}$: Set of notes in score $\mathfrak{S}$ with cardinality N_x
  - $\boldsymbol{\mathcal{O}}$: Set of onsets in score $\mathfrak{S}$ with cardinality N_o (we assume that the onsets are ordered by score position)
- $\mathbf{T}$: Performance representation
- BP_{ave}: Average beat period

Output: Information to reconstruct MIDI file:

- `Pitch`: MIDI pitch of each note in the score
- `Velocity`: Performed MIDI velocity of each note in the score
- `Onset`: Performed onset time in seconds of each note in the score
- `Duration`: Performed duration in seconds of each note in the score

for $0 \leq i \leq N_o - 1$ **do**

Compute the beat period for score onset o_i as

$$\text{BP}(o_i) = \frac{\text{BP}_{ave}}{|o_i|} \sum_{k \in o_i} 2^{t_{k;\log \text{bpr}}}, \tag{3.12}$$

where $t_{k;\log \text{bpr}}$ is the component in the k-th row of $\mathbf{T}$ corresponding to the log BPR defined in Equation (3.8). We slightly abuse notation and denote $k \in o_i$ as the indices corresponding to the notes belonging to score onset o_i.

if $i = 0$ **then**

Initialize the first equivalent onset

$$\hat{o}_0^{perf} = 0 \tag{3.13}$$

else

Compute the equivalent onset

$$\hat{o}_i^{perf} = \hat{o}_{i-1}^{perf} + \text{BP}(o_i) \times \text{IOI}_i^{score}, \tag{3.14}$$

where IOI_i^{score} is the score onset corresponding to onset o_i, as defined in Equation (3.5) (see Section 3.3.1).

Initialize vectors for storing information of the MIDI performance

$$\texttt{Pitch} = \mathbf{0} \in \mathbb{Z}^{N_x} \qquad \texttt{Velocity} = \mathbf{0} \in \mathbb{Z}^{N_x} \qquad \texttt{Onset} = \mathbf{0} \in \mathbb{R}^{N_x} \qquad \texttt{Duration} = \mathbf{0} \in \mathbb{R}^{N_x}$$

for $0 \leq i \leq N_x - 1$ **do**

Get MIDI pitch of score note n_i

$$\texttt{Pitch}_i = \text{pitch}(n_i) \tag{3.15}$$

Compute MIDI velocity of score note n_i

$$\texttt{Velocity}_i = 127 \times t_{i;\text{vel}}, \tag{3.16}$$

where $t_{i;\text{vel}}$ is the component in the i-th row of $\mathbf{T}$ corresponding to the expressive parameter encoding MIDI velocity defined in Equation (3.3).

Compute the onset time for score note n_i

$$\texttt{Onset}_i = \hat{o}^{perf}(n_i) - t_{i;\text{tim}} \tag{3.17}$$

where $t_{i;\text{tim}}$ is the component in the i-th row of $\mathbf{T}$ corresponding to the timing parameter defined in Equation (3.9), and $\hat{o}^{perf}(n_i)$ is the equivalent performed onset corresponding to the score onset to which n_i belongs.

Compute duration for score note n_i

$$\texttt{Duration}_i = 2^{t_{i;\log \text{art}}} \times \text{duration}(n_i) \times \text{BP}(o(n_i)), \tag{3.18}$$

where $t_{i;\log \text{art}}$ is the component in the i-th row of $\mathbf{T}$ corresponding to the log articulation ratio defined in Equation (3.10) and $\text{BP}(o(n_i))$ is the beat period corresponding to the score onset to which n_i belongs.

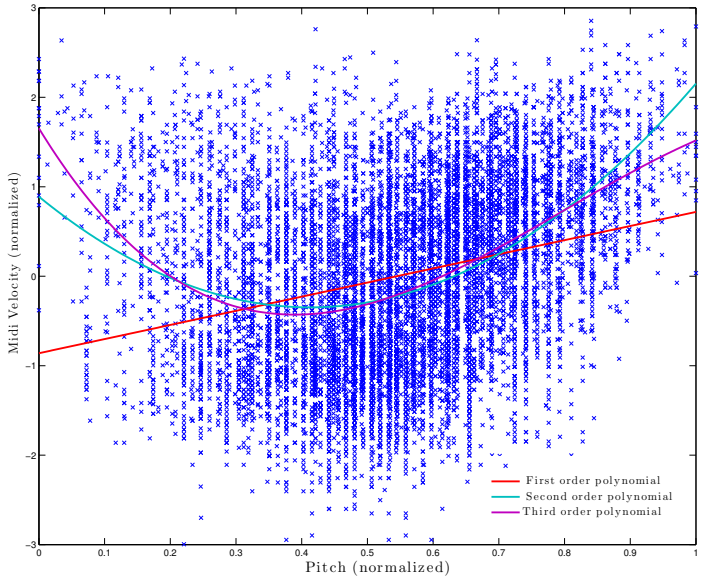


Figure 3.10: Motivation for the polynomial pitch model: polynomial fitting of the Magaloff/Chopin dataset.

loudness levels. We distinguish between three types of dynamics markings: *indicator* (e.g. ***sf***), represented by impulse functions, incremental (e.g. ***crescendo***, illustrated in Figure 3.9), represented by a ramp function and constant (e.g. ***p*** and ***f***, illustrated in Figure 3.9), described by step functions.

3. **Vertical neighbors**. Two basis functions that evaluate the number of simultaneous notes with lower, and higher pitches, respectively, and a third basis function that evaluates the total number of simultaneous notes at that position.

4. **IOI**. For all notes in score onset o_i, a total of six basis functions represent the score IOIs between the three previous onsets and the next three onsets, i.e., the onsets between $(i-2, i-3)$, $(i-1, i-2)$, $(i, i-1)$, $(i, i+1)$, $(i+1, i+2)$, and $(i+2, i+3)$. These basis functions provide some context of the (local) rhythmical structure of the music.

5. **Ritardando**. Encoding of markings that indicate gradual changes in the tempo of the music; includes functions for ***rallentando***, ***ritardando***, ***accelerando***.

6. **Slur**. A representation of *legato* articulations indicating that musical notes are performed smoothly and connected, i.e. without silence between each note. The beginning and ending of a slur are represented by decreasing and increasing ramp functions, respectively. The first (denoted *slur decr*) ranges from one to zero, while the second (denoted *slur incr*) ranges from zero to one over the course of the slur.

7. **Duration**. A basis function that encodes the notated duration of a note.

8. **Rest**. Indicates whether notes precede a rest.

9. **Repeat**. Takes into account repeat and ending barlines, i.e. explicit markings that indicate the structure of a piece by indicating the end of a particular section (which can be repeated), or the ending of a piece. The barlines are represented by an anticipating ramp function leading up to the repeat/ending barline over the course of a measure.

10. **Accent**. Accents of individual notes or chords, such as the ***marcato*** in the example in Figure 3.9.
11. **Staccato**. Encodes ***staccato*** markings on a note, an articulation indicating that a note should be temporally isolated from its successor, by shortening its duration.
12. **Grace notes**. Encoding of musical ornaments that are melodically and/or harmonically nonessential, but have an embellishment purpose.
13. **Fermata**. A basis function that encodes markings that indicate that a note should be prolonged beyond its normal duration.

The right side Figure 3.9 shows the score representation $\mathbf{\Phi}$ of our running example for a performance representation model consisting of basis functions describing pitch, dynamics markings ***p***, ***f*** and ***crescendo***; and a basis function encoding an accent (a ***marcato***).

3.3.3 Model Description

We can model the value of an expressive parameter for score note n_i as a linear combination of the input basis functions, i.e.

$$f_{lbm}(n_i\,;\mathbf{w}) = \mathbf{w}^{\mathrm{T}}\boldsymbol{\varphi}(n_i), \tag{3.19}$$

where $\mathbf{w} \in \mathbb{R}^M$ is a vector of weights. The complete predictive function for all expressive parameters defined in Section 3.3.1 can be written as

$$F_{lbm}(n_i) = \begin{pmatrix} f_{lbm}^{(\mathrm{vel})}(\boldsymbol{\varphi}(n_i)\,;\mathbf{w}_{\mathrm{vel}}) \\ f_{lbm}^{(\log \mathrm{bpr})}(\boldsymbol{\varphi}(n_i)\,;\mathbf{w}_{\log \mathrm{bpr}}) \\ f_{lbm}^{(\mathrm{tim})}(\boldsymbol{\varphi}(n_i)\,;\mathbf{w}_{\mathrm{tim}}) \\ f_{lbm}^{(\log \mathrm{art})}(\boldsymbol{\varphi}(n_i)\,;\mathbf{w}_{\log \mathrm{art}}) \end{pmatrix} = \begin{pmatrix} \mathbf{w}_{\mathrm{vel}}^{\mathrm{T}}\boldsymbol{\varphi}(x_i) \\ \mathbf{w}_{\log \mathrm{bpr}}^{\mathrm{T}}\boldsymbol{\varphi}(n_i) \\ \mathbf{w}_{\mathrm{tim}}^{\mathrm{T}}\boldsymbol{\varphi}(n_i) \\ \mathbf{w}_{\log \mathrm{art}}^{\mathrm{T}}\boldsymbol{\varphi}(n_i) \end{pmatrix}, \tag{3.20}$$

where the set of parameters is $\mathbf{\Theta} = \{\mathbf{w}_{\mathrm{vel}}, \mathbf{w}_{\log \mathrm{bpr}}, \mathbf{w}_{\mathrm{tim}}, \mathbf{w}_{\log \mathrm{art}}\}$.

It is easy to see that the above definition of the LBM models expressive parameters independently: the prediction of the parameter encoding MIDI velocity for score note n_i does not depend on the prediction of the log BPR for that note. Furthermore, the above formulation of the LBM implicitly assumes that the prediction of the values of an expressive parameter for different score notes are also independent of each other[13]: the prediction of MIDI velocity for score note n_i only depends on the score information for that note (i.e. $\boldsymbol{\varphi}(n_i)$). While these independence assumptions simplify the mathematical description of the model, they are not necessarily musically plausible.

In the following discussion, we will refer to a model as *sequential* if its predictions consider the temporal context of the music (i.e. if it considers the sequential nature of music), and *non-sequential* its predictions are independent of each other. In this case, the LBM described above is an example of a non-sequential model.

[13] It can be shown that the deterministic linear model described in this section is equivalent to a probabilistic model, which assumes that the conditional distribution of the value of an expressive parameter for score note n_i, given the score information represented by $\boldsymbol{\varphi}(n_i)$, is a Gaussian distribution centered around $f_{lbm}(n_i\,;\mathbf{w})$, and that the values of the expressive parameters for each note are independent and identically distributed (i.i.d). See Appendix E.1 for a detailed derivation of this result.

Training

Since the LBM models expressive parameters independently, we optimize each set of weights separately. To unclutter notation, in the following we will simply use $\mathbf{w}$ to refer to the parameters of f, the predictive function modeling an expressive parameter (i.e. the component of the full predictive function F_{lbm} defined in Equation (3.20) that models the expressive parameter we are optimizing the model for), since the following procedure is identical for each $\mathbf{w} \in \boldsymbol{\theta}$. Given a training set $\boldsymbol{\mathcal{T}}$ the parameters of the predictive function corresponding to an expressive parameter (MIDI velocity, log BPR, timing or log articulation) can be learned by minimizing the squared error, given by

$$\mathcal{L}_{se}(\mathbf{w}) = \|\mathbf{t}_{\boldsymbol{\mathcal{T}}} - \boldsymbol{\Phi}_{\boldsymbol{\mathcal{T}}}\mathbf{w}\|^2 \tag{3.21}$$

where $\mathbf{t}_{\boldsymbol{\mathcal{T}}}$ is a vector concatenating the values of the expressive parameter for all pieces in the test set (i.e. the columns of each performance representation $\mathbf{T}_i \in \boldsymbol{\mathcal{T}}$ corresponding to the expressive parameter that we are optimizing the parameters of the predictive function for); and $\boldsymbol{\Phi}_{\boldsymbol{\mathcal{T}}}$ is a matrix concatenating the score features of each score in the training set. These quantities are given by

$$\mathbf{t}_{\boldsymbol{\mathcal{T}}} = \begin{pmatrix} \mathbf{t}_1 \\ \vdots \\ \mathbf{t}_{N_p} \end{pmatrix} \in \mathbb{R}^{N_T} \qquad \boldsymbol{\Phi}_{\boldsymbol{\mathcal{T}}} = \begin{pmatrix} \boldsymbol{\Phi}_1 \\ \vdots \\ \boldsymbol{\Phi}_{N_p} \end{pmatrix} \in \mathbb{R}^{N_T \times M} \tag{3.22}$$

where N_T is the total number of notes in all scores in $\boldsymbol{\mathcal{T}}$. In the machine learning literature, this problem is known as *least squares* (LS). The least squares solution can be computed analytically as

$$\mathbf{w}_{ls} = \underset{\mathbf{w}}{\operatorname{argmin}}\, \mathcal{L}_{se}(\mathbf{w}) = \boldsymbol{\Phi}^{\dagger}_{\boldsymbol{\mathcal{T}}}\mathbf{t}_{\boldsymbol{\mathcal{T}}}, \tag{3.23}$$

where $\boldsymbol{\Phi}^{\dagger}_{\boldsymbol{\mathcal{T}}} = \left(\boldsymbol{\Phi}^{\mathrm{T}}_{\boldsymbol{\mathcal{T}}}\boldsymbol{\Phi}_{\boldsymbol{\mathcal{T}}}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}}_{\boldsymbol{\mathcal{T}}}$ is the *Moore-Penrose pseudo-inverse* of $\boldsymbol{\Phi}_{\boldsymbol{\mathcal{T}}}$. A detailed derivation of this result is provided in Appendix E.1. The computation of $\boldsymbol{\Phi}^{\dagger}_{\boldsymbol{\mathcal{T}}}$ might result in numerical instabilities for ill conditioned problems[14], which means that the estimation of the parameters $\mathbf{w}_{ls}$ might be extremely sensitive to small changes in $\mathbf{t}_{\boldsymbol{\mathcal{T}}}$. In other words, the parameters of the predictive function might be very different for two very similar (but not identical) training sets. Furthermore, for large training datasets, the exact solution to an LS problem can be computationally expensive (Bishop, 2006). More efficient alternatives for computing the parameters include the LSQR and LSMR algorithms for large sparse systems (Paige and Saunders, 1982; Fong and Saunders, 2011).

3.4 Bayesian Linear Basis Models

Although the formulation of the LBM using least squares described in Section 3.3.3 is deterministic, it can be shown to be equivalent to a probabilistic approach assuming that the expressive parameters are normally distributed[15], and that we are only interested in the most likely solution (i.e. the expected value). In this section we describe a fully probabilistic extension of the

[14] From a more technical perspective, the problem of computing the pseudo-inverse of a matrix is said to be ill conditioned, if the ratio of the largest to the smallest singular value of the matrix is much larger than 1. This can be an issue for large sparse matrices, like the ones generated by the score representation models described in this work.

[15] See Appendix E.1.

LBM which allows for incorporating prior knowledge in the form of prior distributions over the model parameters using a Bayesian approach, which we will refer to as Bayesian LBM (BLBM). Such a probabilistic approach is an important step in order to account for the fact that there may be multiple distinct ways to perform music. This formulation of the BM approach was first introduced by Grachten et al. (2014).

3.4.1 Basis Functions

As previously mentioned, this thesis documents the historical development of the BM framework. In addition to the basis functions described in Section 3.3.2, for this version of the BM framework, we define a new group of basis functions that incorporate contextual information about dynamics markings:

1. **Context Dynamics Markings**. This set of basis functions intends to differentiate between gradual loudness annotations in different loudness contexts. We do this by combining each gradual annotation with its preceding and succeeding loudness level, for example ***p*** → ***crescendo*** → ***mf***, or ***f*** → ***diminuendo*** → ***mf***.

Note that these context dynamics markings can define a very large number of basis functions, which might lead to very sparse score representations. For a formal description of these basis functions see Appendix A.1.

3.4.2 Model

Using the chain rule of probability, the joint probability distribution of an expressive parameter $\mathsf{y}_j(n_i) = t_{ij}$, the value of the j-th expressive parameter (MIDI velocity, log BPR, timing or log articulation) for score note n_i, and the parameters $\boldsymbol{\theta}_j$ of the predictive function for the j-th expressive parameter can be written as

$$p(t_{ij}, \boldsymbol{\theta}_j) = p(t_{ij} \mid \boldsymbol{\theta}_j) p(\boldsymbol{\theta}_j), \tag{3.24}$$

where $p(t_{ij} \mid \boldsymbol{\theta}_j)$ is the conditional probability distribution of expressive parameter t_{ij} given parameters $\boldsymbol{\theta}_j$ and $p(\boldsymbol{\theta}_j)$ is the conjugate prior distribution of parameters $\boldsymbol{\theta}_j$. Using this probabilistic perspective, we can rethink the approach proposed in Equation (3.1) as modeling the conditional distribution of t_{ij} given $\boldsymbol{\theta}_j$, which, assuming a Gaussian error term, can be written as

$$p(t_{ij} \mid \boldsymbol{\theta}_j) = \mathcal{N}(t_{ij} \mid f^{(j)}(\boldsymbol{\varphi}(n_i)\,; \mathbf{w}_j), \beta_j^{-1}), \tag{3.25}$$

where β_j is the precision (i.e. the inverse variance) of the distribution[16] and $\mathbf{w}_j$ are the weights of the j-th predictive function. Using Equation (3.19), we can rewrite the above equation as

$$p(t_{ij} \mid \mathbf{w}_j) = \mathcal{N}(t_{ij} \mid \mathbf{w}_j^{\mathrm{T}} \boldsymbol{\varphi}(n_i), \beta_j^{-1}). \tag{3.26}$$

Since a similar distribution can be defined for each of the expressive parameters, in the rest of this discussion, to unclutter notation we will drop the subscript j (e.g. we will write t_{ij} as t_i).

[16] The discussion on Bayesian linear models provided in this section follows the notation conventions from Section 3.3 in (Bishop, 2006). Following these conventions, we write the conditional distribution in terms of precision since in practice it is more numerically stable to estimate the precision than the variance.

Assuming that the values of the expressive parameter for the whole score $\mathsf{y}(\mathfrak{S}) = \mathbf{t}$ are i.i.d[17], their conditional probability distribution is given as

$$p(\mathbf{t} \mid \mathbf{w}) = \prod_{i=0}^{N_x-1} \mathcal{N}(t_i \mid \mathbf{w}^{\mathrm{T}}\boldsymbol{\varphi}(n_i), \beta^{-1}). \tag{3.27}$$

Using a Bayesian interpretation, we assume that the weights of the predictive function have a prior distribution $p(\mathbf{w}) = \mathcal{N}(\mathbf{w} \mid \mathbf{m}_0, \mathbf{S}_0)$, where $\mathbf{m}_0$ and $\mathbf{S}_0$ are the mean and covariance respectively. Using this prior distribution and the conditional distribution of $\mathbf{t}$ given the weights $\mathbf{w}$ from Equation (3.27), it follows from Bayes' theorem that the posterior probability of the weights $\mathbf{w}$ given the expressive parameters $\mathbf{t}$ is also a Gaussian distribution, i.e.

$$p(\mathbf{w} \mid \mathbf{t}) = \mathcal{N}(\mathbf{w} \mid \mathbf{m}_N, \mathbf{S}_N), \tag{3.28}$$

where the mean and covariance are given respectively by

$$\mathbf{m}_N = \mathbf{S}_N(\mathbf{S}_0^{-1}\mathbf{m}_0 + \beta\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t}) \qquad \text{and} \qquad \mathbf{S}_N^{-1} = \mathbf{S}_0^{-1} + \beta\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}, \tag{3.29}$$

where $\mathcal{F}(\mathfrak{S}) = \boldsymbol{\Phi}$ is the score representation using basis functions.

A common simplification is to assume that the posterior distribution is a zero mean and isotropic Gaussian, i.e.,

$$\mathbf{m}_0 = \mathbf{0} \qquad \text{and} \qquad \mathbf{S}_0 = \alpha^{-1}\mathbf{I}, \tag{3.30}$$

where $\mathbf{0}$ is a vector of zeros, α is the precision of the posterior distribution, and $\mathbf{I}$ is the identity matrix.

As mentioned previously, an advantage of a probabilistic formulation of the BM is that it allows for generating multiple "valid" performances of a piece, i.e. given a trained model, there might be several different performances that can be rendered using the model. The most straightforward way to do so is to define the predictive function of the BLBM as the expected value of the expressive parameter t_i under $p(t_i \mid \mathbf{w})$, i.e.

$$\begin{aligned} f_{(blbm)}(n_i) &= \mathbb{E}\{t_i\} \\ &= \mathbf{w}^{\mathrm{T}}\boldsymbol{\varphi}(n_i), \end{aligned} \tag{3.31}$$

which has the same form as the LBM and thus, the complete predictive function has the same form as Equation (3.20). Alternatively, we could generate performances by sampling from the predictive conditional distribution. Some techniques for generating performances in this way are reviewed in Appendix G.

Training Similar to the case of the LBM, the BLBM models expressive parameters independently, and thus, training the model involves training each set of parameters separately. Given a training set $\boldsymbol{\mathcal{T}}$, the model parameters can be learned using the *maximum a posteriori* (MAP) principle, which minimizes the negative of the posterior log-likelihood of the weights of the predictive function $\mathbf{w}$ given the expressive parameters in the training set $\mathbf{t}_{\boldsymbol{\mathcal{T}}}$, defined in Equation (3.22), given by

$$\mathcal{L}_{map}(\mathbf{w}) = -\log p(\mathbf{w} \mid \mathbf{t}_{\boldsymbol{\mathcal{T}}}). \tag{3.32}$$

[17] As discussed in Section 3.3.3, this independence assumption is musically unlikely.

In the simplified case for the zero mean isotropic covariance, it is possible to compute a closed form analytical solution, given by

$$\mathbf{w}_{map} = \underset{\mathbf{w}}{\operatorname{argmin}}\, \mathcal{L}_{map}(\mathbf{w}) = \left(\frac{\alpha}{\beta}\mathbf{I} + \mathbf{\Phi}_{\boldsymbol{\mathcal{T}}}^{T}\mathbf{\Phi}_{\boldsymbol{\mathcal{T}}}\right)^{-1} \mathbf{\Phi}_{\boldsymbol{\mathcal{T}}}^{\mathrm{T}}\mathbf{t}_{\boldsymbol{\mathcal{T}}}. \tag{3.33}$$

For a full derivation of this result, see Appendix E.3. The hyper-parameters α and β can be computed using the *evidence approximation* algorithm (Bishop, 2006).

From the above equation is easy to see that, if we assume that the prior distribution of parameters $\mathbf{w}$ of the predictive function to be a zero mean isotropic Gaussian distribution, the Bayesian linear regression is equivalent to a regularized LS regression[18], with a regularization coefficient of $\frac{\alpha}{\beta}$. Therefore, the Bayesian linear regression is less prone to overfitting. Furthermore, this similarity with regularized LS provides us with an efficient way to estimate the parameters $\mathbf{w}_{map}$: first estimate the precisions α and β, and then compute the regularized LS solution using an iterative method for large sparse systems, such as LSQR (Paige and Saunders, 1982).

In the case of $\|\mathbf{S}_0^{-1}\| \to 0$ (or $\alpha \to 0$), the weights $\mathbf{w}_{map}$ converge to the LS weights $\mathbf{w}_{ls}$. In such a scenario, the prior probability distribution is said to be *non informative* (Bishop, 2006).

Algorithm E.1 in Appendix E.4 presents an iterative procedure for computing the MAP solution in the general case for arbitrary mean and covariance using the *expectation–maximization* (EM) algorithm (Dempster et al., 1977).

3.5 Evaluation

In this section, we present three quantitative experiments for evaluating the LBM and BLBM models discussed above.

The first and second experiments compare how well the two versions of the linear BMs discussed in this chapter (the LBM and the BLBM) account for aspects of expressive dynamics (encoded by the MIDI velocity, as defined in the performance representation model described in Section 3.3.1). In the first experiment, we assess how accurately expressive dynamics can be represented by the BMs (in a goodness of fit scenario). The second experiment compares the flexibility of the LBM and BLBM to obtain good predictions for new pieces (i.e. pieces not included in the training set) by means of a leave-one-out cross validation. The third experiment explores the predictive accuracy of the BLBM for all expressive parameters defined in performance codec v. 1.0 defined in Section 3.3.1. These experiments were an important means of determining which variant of the predictive model (LS linear model vs. Bayesian linear model) produces more accurate reconstructions of human performances. They were also important for identifying which groups of basis functions contribute to more accurate performance predictions.

For the experiments reported in this chapter, we use the Magaloff/Chopin dataset described in Section 2.4. As previously stated, this dataset consists of performances of the complete piano solo works by F. Chopin performed by Nikita Magaloff.

We use two metrics to quantitatively evaluate the predictions of the model: the *sample Pearson correlation coefficient* and R^2, the *coefficient of determination*. The correlation coefficient denotes how strongly the observed expressive parameters and the values of the expressive parameters predicted by the model correlate. For a piece with observed parameters $\mathsf{y}(\mathfrak{S}) = \mathbf{t}$ and

[18] See Appendix E.2 for a detailed derivation of the solution of regularized LS.

model predictions $f(\mathbf{\Phi}\,;\mathbf{w}) = \mathbf{y}$, the Pearson correlation coefficient is computed as

$$r = \frac{\sum_{i=1}^{N}(t_i - \bar{t})(y_i - \bar{y})}{\sqrt{\sum_{i}^{N}(t_i - \bar{t})^2}\sqrt{\sum_{i}^{N}(y_i - \bar{y})^2}}, \tag{3.34}$$

where $\bar{t}$ and $\bar{y}$ are the sample mean (i.e. the average values) of the observed parameters $\mathbf{t}$ and the predicted parameters $\mathbf{y}$, respectively.

The coefficient of determination R^2 expresses the proportion of variance explained by the model. This quantity is defined as

$$R^2 = 1 - \frac{SS_{err}}{SS_t}, \tag{3.35}$$

where SS_{err} is the sum of squared errors between the predictions and the expressive parameters, and SS_t is the total sum of squares of the expressive parameter. The R^2 measure has an upper bound of 1, and has no lower bound (predictions can be arbitrarily far away from the target values). Positive R^2 values indicate that the models perform better than the baseline $R^2 = 0$, which can be achieved by predicting the average value of the observed expressive parameter.

The rest of this section is structured as follows: Section 3.5.1 describes the first experiment evaluating the goodness of fit of the model for expressive dynamics. Section 3.5.2 describes the leave-one-out cross validation experiment evaluating the predictive accuracy for expressive dynamics. Section 3.5.3 describes the leave-one-out cross validation experiment evaluating the predictive accuracy of all expressive parameters.

In all of the experiments described above, we use the zero mean isotropic Gaussian version of the BLBM (see Equation (3.30)). The predictions of the BLBM are generated taking the expected value of the conditional distribution, using Equation (3.31).

3.5.1 Goodness of fit: Expressive Dynamics

The main objective of the first experiment was to compare how accurately an expressive parameter, in particular, expressive dynamics encoded by the MIDI velocity (as described in Section 3.3.1), could be represented and predicted using both versions of the linear BMs discussed in this chapter.

For this experiment, we used a selection of four pieces from the Magaloff/Chopin corpus. As mentioned in Section 2.4, the scores in this dataset were compiled using optical music recognition. Since this process is not perfect, manual corrections were required to ensure that the score information was properly represented. The scores of the four pieces selected for this experiment were manually corrected, in particular the dynamics markings. This corrected subset consisted of Nocturnes Op. 15 No. 1 and Op. 27 No. 2, Prelude Op. 28 No. 17, and Ballade Op. 52.

In this experiment we were also interested in investigating which groups of basis functions contribute to more accurate model predictions. We tested the following feature sets: PIT, which consisted of the three basis functions in the polynomial pitch model; DYN, which consisted of the basis functions described in Section 3.3.2, without the polynomial pitch model; and DYN_c, which consisted of the same basis functions as DYN, but substituted the basis functions for dynamics markings described in Section 3.3.2 by the context dynamics basis functions described in Section 3.4.1. DYN + PIT refers to the combined features included in DYN and PIT, and DYN_c + PIT refers to the combined features included in DYN_c and PIT.

	LBM		BLBM	
Feature Set	r	R^2	r	R^2
DYN	0.516	0.275	0.516	0.276
DYN_c	0.576	0.339	0.576	0.339
PIT	0.416	0.174	0.416	0.174
DYN+PIT	0.622	0.393	0.621	0.393
DYN_c + PIT	0.675	0.462	0.675	0.461

Table 3.1: Goodness-of-fit of the model over performances of four Chopin piano pieces. See Section 3.5 for abbreviations.

This experiment was conducted as follows: Each model (LBM and BLBM) was trained using each of the feature sets described above on all four (manually corrected) pieces, for a total of 10 evaluations.

After the models were trained, they were used to make predictions of the performed dynamics for the same four pieces. We evaluated the predictions for each piece using r and R^2 and average across pieces for each features set/model combination. Table 3.1 shows these averaged r and R^2 values.

Table 3.1 shows the average correlation coefficient r and coefficient of determination R^2 over the four pieces for the LS approximation, as well as for the Bayesian approach.

3.5.2 Predictive Accuracy: Expressive Dynamics

For the second experiment we were interested in evaluating the predictive accuracy of the LBM and BLBM models for expressive dynamics. We tested how accurately the models predicted dynamics for new pieces that were not included in the training set. Furthermore, in this experiment we also aimed to identify the subset of basis functions that led to better predictions.

We tested the predictive accuracy by means of 10 leave-one-out cross validation experiments, using 151 pieces from the Magaloff/Chopin dataset.

Each model was trained using one of the feature sets described in the previous section (DYN, PIT, etc.). This training used all pieces but one, and predictions were subsequently made for that excluded piece. The same procedure was performed for all pieces in the 151-piece dataset, such that each piece in the dataset was predicted once.

Table 3.2 shows the accuracy of the model in this scenario. As in the case of goodness of fit, we used r and R^2 to evaluate the predictive accuracy of the models. These values were averaged across pieces for each feature set/model combination.

3.5.3 Predictive Accuracy: All Expressive Parameters

The third experiment used the BLBM and DYN + PIT feature set to predict all of the expressive parameters defined in performance codec v. 1.0 (see Section 3.3.1), including MIDI velocity, log BPR, timing, and log articulation. The aim was to test whether this model/feature set combination, which proved most successful for predicting expressive dynamics, would also accurately predict other expressive parameters.

	LBM		BLBM	
Feature Set	r	R^2	r	R^2
DYN	0.181	0.073	0.181	0.073
DYN_c	0.185	0.072	0.185	0.072
PIT	0.381	0.166	0.381	0.166
DYN+PIT	0.431	0.207	0.431	0.207
DYN_c + PIT	0.420	0.198	0.419	0.198

Table 3.2: Predictive accuracy in a leave-one-out scenario over performances of 151 Chopin piano pieces. See Section 3.5 for abbreviations.

Expressive Parameter	r	R^2
MIDI velocity	0.431	0.207
log BPR	0.200	0.035
Timing	0.178	-0.107
log Articulation	0.313	0.096

Table 3.3: Predictive accuracy of the Bayesian linear regression using feature set DYN + PIT for the expressive parameters defined in performance representation model v. 1.0, described in Section 3.3.1.

This experiment tested predictive accuracy using 4 leave-one-out cross validations, each corresponding to an expressive parameter, and used the same 151 pieces from the Magaloff/Chopin dataset as the second experiment.

Table 3.3 shows r and R^2, averaged across pieces for each expressive parameter.

As an illustration of predictive accuracy, Figure 3.11 shows a comparison of the predictions and the expressive parameters for Chopin's Prelude Op. 28 No. 11 in B Major.

3.6 Discussion

The results for both goodness of fit and predictive accuracy of expressive dynamics show that the Bayesian approach (BLBM) performs on par with the standard LS regression (LBM). For the goodness of fit experiment (Table 3.1), an independent-samples two-tailed t-test was conducted to compare differences in R^2 between LBM and BLBM models. There was no significant difference at the $p = 0.01$ level between LBMs ($mean = 1.64, std = 0.22$) and BLBMs ($mean = 1.64, std = 0.22$); $t(38) = 0.00, p = 1.0$.

A similar test was performed for the results of the predictive accuracy experiment (Table 3.2). For these results there was no significant difference at the $p = 0.01$ level between LBMs ($mean = 0.72, std = 0.74$) and BLBMs ($mean = 0.72, std = 0.22$); $t(1508) = 0.00, p = 1.0$. During the realization of the experiments, we noted that the hyper-parameter α (the precision of the prior distribution of the weights of the predictive function $p(\mathbf{w})$) tends to be very small, which suggests that the prior probability is non-informative. Therefore the assumption that the priors have a centered unimodal distribution, as discussed in Section 3.4.2, could be an oversimplification.

We can also see that the use of more sophisticated basis functions for encoding dynamics markings (the DYN_c feature set described above) does not increase the predictive accuracy.

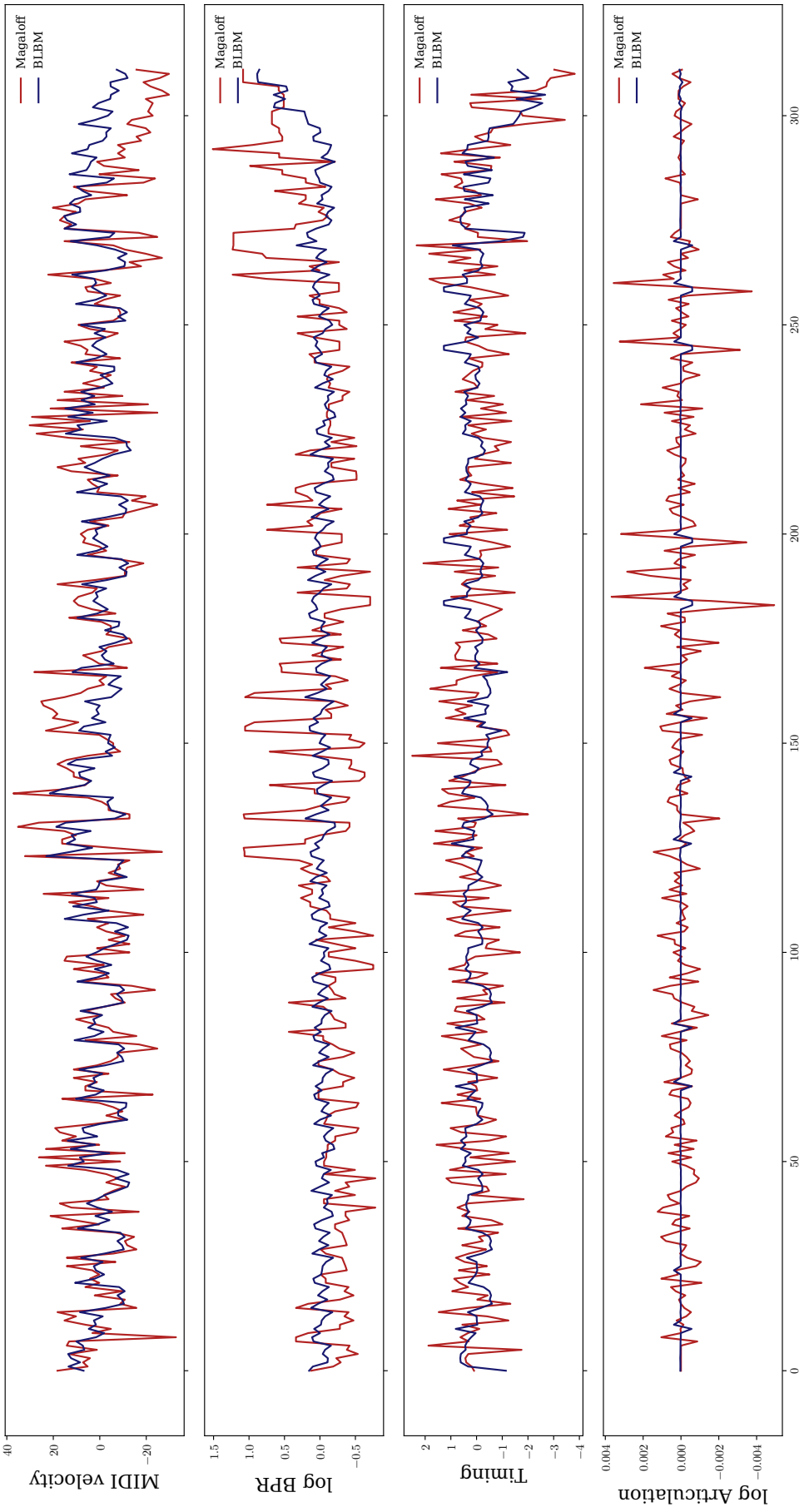


Figure 3.11: Comparison of the predictions of the expressive parameters made by the BLBM (in blue) Magaloff's performance (in red) for Prelude in B major Op. 28, No. 11 by F. Chopin.

For the goodness of fit experiment, an independent-samples two-tailed t-test was conducted to compare models trained on DYN and DYN_c. This test showed a significant difference at the $p = 0.01$ level between DYN ($mean = 1.34, std = 0.22$) and DYN_c ($mean = 1.60, std = 0.23$); $\text{t}(30) = -3.36, p = 0.002$. Nevertheless, the analogous test conducted for the predictive accuracy experiment did not show a significant difference between DYN ($mean = 0.56, std = 0.46$) and DYN_c ($mean = 0.54, std = 0.48$); $\text{t}(1206) = 0.74, p = 0.460$. A possible explanation for this non-significant finding could be a lack of sufficient training data to represent all possible basis functions. Alternatively, the significant effect shown for goodness of fit might be due to overfitting: the over-modeling provided by the basis functions DYN_c, presents an increase of ca. 12% in the correlation coefficient, and almost 22% more explained variance when compared to the more general basis DYN.

The results in Table 3.3 show that the BLBM may work better for predicting expressive dynamics than other expressive parameters. These results suggest that a linear approach that does not consider the performance context (i.e., disregards the sequential nature of music) might not be appropriate for modeling expressive tempo and timing. This contrast in predictive accuracy is illustrated in Figure 3.11: for example, the predictions for log BPR and timing do not capture the variance present in the human performance. Another factor to consider is that pitch-related features seem to be particularly relevant for predicting dynamics (as suggested by the results of PIT in Tables 3.1 and 3.2), implying that different expressive parameters may require different types of basis functions.

3.7 Conclusions

This chapter is the first of a two-part comprehensive and formal description of the BM framework for expressive performance. The BM framework was presented as a formalization of the components of the general computational model of expressive performance described in Section 2.3. We presented a formalization of the components of a BM: basis functions representing score information, expressive parameters representing performance information, and predictive functions relating score and performance information. We then introduced two versions of the linear BM: the original deterministic approach, introduced by Grachten and Widmer (2012), and a probabilistic approach that uses a Bayesian framework introduced by Grachten et al. (2014).

We found the Bayesian model to perform on par with the LS regression. As discussed above, the lack of improvement shown by the Bayesian model suggests that the Gaussian assumptions on the prior distribution of the parameters might be an oversimplification. The use of the set of basis functions that encodes dynamics markings in a context-aware fashion, although musically justifiable, failed to improve predictive accuracy. An explanation for this is that the context-aware representation of the annotations creates a higher-dimensional, and thus less densely populated data space, in which it is harder to generalize to unseen data.

The predictions of the linear models work best for expressive dynamics, suggesting that other expressive parameters might require more sophisticated regression models and different types of basis functions.

The linear BM also has some important shortcomings. Firstly, the model can only learn linear relationships between the basis functions and the expressive parameters. A second shortcoming in linear models is that basis functions are assumed to influence expressive parameters independently of all other basis functions. In reality, it is conceivable that, for example, the effect of one particular basis function on expressive timing is different depending on which other basis

functions are active at the same time. Third, linear models do not consider the sequential nature of music.

The following chapters will address each of these shortcomings. Chapter 4 begins with an exploration of whether the same basis function framework benefits from a more powerful, non-linear model. The second part of the chapter focuses on sequential modeling. Chapter 5 expands the model to the case of ensemble performance, and Chapter 6 tests whether cognitively plausible features improve predictions of expressive dynamics and tempo.

4 Basis Function Models II: Non-linear Models

This chapter contains material published in

- Cancino Chacón, C. E. and Grachten, M. (2015). An Evaluation of Score Descriptors Combined with Non-linear Models of Expressive Dynamics in Music. In *Proceedings of the 18th International Conference on Discovery Science (DS 2015)*, pages 48–62, Banff, AB, Canada
- Cancino-Chacón, C. E., Gadermaier, T., Widmer, G., and Grachten, M. (2017d). An Evaluation of Linear and Non-linear Models of Expressive Dynamics in Classical Piano and Symphonic Music. *Machine Learning*, 106(6):887–909
- Grachten, M. and Cancino-Chacón, C. E. (2017). Temporal dependencies in the expressive timing of classical piano performances. In Lessafre, M., Maes, P.-J., and Leman, M., editors, *The Routledge Companion to Embodied Music Interaction*, pages 360–369. Routledge

4.1 Introduction

In Chapter 3 we introduced the basis function modeling (BM) framework for musical expression. In its simplest form, a linear BM just expresses the value of an expressive parameter at a particular position as a linear combination of the basis functions that are non-zero at that position. In this setup, each basis function has a positive or negative weight parameter that determines how strongly (and in what direction) that basis function influences the expressive parameter.

As discussed in Chapter 3, the advantages of such linear models are firstly that it is easy to find optimal weight parameters given a set of performances (using the standard least squares (LS) regression or the Bayesian linear regression described in Sections 3.3.3 and 3.4.2, respectively), and secondly that it is straightforward to tell what the model has learned from the data by looking at the weight parameters. For example, a positive weight for the *fermata* basis function (which indicates the presence of a fermata in the score) shows that the model has learned that a fermata sign causes an increase in duration (that is, a lengthening of the note at the fermata).

Although the linear BMs produce surprisingly good results given their simplicity, as discussed in Section 3.7, they also have some important shortcomings. Firstly, they can only learn linear relationships between the basis function and the expressive parameter. For instance, in the case of modeling the intensity of notes in a performance, it has been observed that the higher the note, the louder it is played (Friberg et al., 2006; Grachten and Widmer, 2012). However, this relationship is not strictly linear, but roughly follows an S-shaped curve. In that case a basis function that returns the pitch of a note will help to predict the intensity of the note, but it will tend to overestimate the effect of the *higher–louder/lower–softer* effect for the lowest and highest pitches. A second shortcoming in linear models is that they assume that each of the basis functions influences expressive parameters independently of all other basis functions. In reality, it is conceivable that the effect of one particular basis function on an expressive

parameter is different depending on which other basis functions are active at the same time. A third shortcoming is that linear models are non-sequential[1]: by assuming that the performance of each note is independent from the other notes (assumed both by the LS formulation of the original LBM discussed in Section 3.3 and the Bayesian LBM discussed in Section 3.4), linear models do not consider the sequential nature of music.

In the first part of this chapter we explore whether the same basis function framework can benefit from more powerful, non-linear models: feed forward neural networks (FFNNs). In the second part of the chapter we extend this non-linear model to allow modeling the music sequentially by using recurrent neural networks (RNNs). In the rest of this thesis, we will refer to these models as the non-linear basis function model (NBM) and the recurrent non-linear basis function model (RNBM), respectively. Although there are many ways to model non-linear relationships, artificial neural networks (ANNs) offer a flexible and conceptually simple approach, that has proven its merits over the past decades (Bishop, 1995; Goodfellow et al., 2016).

The rest of this chapter is structured as follows: Section 4.2 describes NBMs, non-linear extension of the LBMs using FFNNs, including a quantitative evaluation of the predictive accuracy of these models for expressive dynamics. Section 4.3 describes RNBMs, sequential extension of the NBMs model using RNNs, including an evaluation of their predictive accuracy of for expressive tempo. Finally, Section 4.4 concludes the chapter.

4.2 Non-Linear Basis Models

The first model discussed in this chapter, the NBM, is an extension of the LBM that replaces the linear model with a more powerful non-linear (but also non-sequential) model. In this case, the influence of the basis functions in the expressive parameter can be modeled in a non-linear way using FFNNs. NBMs are note-wise models that predict a value of the expressive parameters for each note in the score, and thus, their expressive parameters are the same as the ones defined in the performance codec $\mathcal{C}_{1.0}$ described in Section 3.3.1: MIDI velocity, log BPR, timing and log articulation. The NBM was introduced in (Cancino Chacón and Grachten, 2015).

Following the notation conventions from Chapters 2 and 3, $\mathfrak{P}_{\mathfrak{S}}$ denotes an expressive performance matched to score $\mathfrak{S}$; $\boldsymbol{\mathcal{X}} = \{n_1, \dots, n_{N_x}\}$ represents the set of $|\boldsymbol{\mathcal{X}}| = N_x$ notes in a musical score and $\boldsymbol{\mathcal{O}} = \{o_1, \dots, o_{N_o}\}$ represents the set of $|\boldsymbol{\mathcal{O}}| = N_o$ unique score positions (score onsets). The set of all notes that occur at the same score position as score note n_i will be denoted as $o(n_i) = \{n_j \in o \mid \text{onset}(n_j) = \text{onset}(n_i)\}$, with $\text{onset}(n_i)$ being the onset time of n_i. An arbitrary element of the score (note or onset) is denoted as x.

4.2.1 Basis Functions

At this point, we remind the reader that this thesis documents the historical development of the BM framework. The experience gained conducting the experiments described in Section 3.5 led to the definition of new basis functions capturing aspects of the metrical structure, as well as to dropping the use of basis functions describing contextual dynamics markings, which are defined in Section 3.4.1. Furthermore, we simplify the polynomial pitch model to only a linear term, since the non-linear models described in this chapter are more powerful. In addition to the basis functions described in Chapter 3, we introduce two new groups of basis functions capturing more aspects of the score:

[1] See Section 3.3.3.

1. **Metrical**. Representation of the time signature of a piece, and the (metrical) position of each note in the bar. For each time signature $\frac{a}{b}$, there are $a+1$ basis functions: a basis functions indicate notes starting at each beat, respectively, and a single basis function indicates notes starting on a *weak* metrical position (off-beat). For example, the basis function labeled $\frac{4}{4}$ *beat 1* evaluates to 1 for all notes that start on the first beat in a $\frac{4}{4}$ time signature, and to 0 otherwise. Although in (Western) music, most time signatures have common accentuation patterns, we choose not to hard-code these accentuation patterns in the form of basis functions. Instead, by defining the metrical basis functions as sets of indicator functions associated to metrical positions, we leave it to the basis models to *learn* the accentuation patterns as they occur in the data.

2. **Harmonic**. Two sets of indicator basis functions that encode a computer–generated harmonic analysis of the score based on the probabilistic polyphonic key identification algorithm proposed in (Temperley, 2007). This harmonic analysis produces an estimate of the key and scale degree, i.e. the roman numeral functional analysis of the harmony of the piece, for each bar of the score. A set of basis functions encode all major and minor keys while another set of basis functions encodes scale degrees.

It should be noted that some of the above mentioned groups of basis functions can comprise a large number of individual basis functions. For example, the traditional binary and ternary time signatures (encoded in the group of Metrical basis functions) generate more than 50 basis functions with the $\frac{12}{8}$ time signature alone generating 13 basis functions, namely $\frac{12}{8}$ *beat 1*, $\frac{12}{8}$ *beat 2*, $\frac{12}{8}$ *beat 3*, ..., $\frac{12}{8}$ *beat 11*, $\frac{12}{8}$ *beat 12*, and $\frac{12}{8}$ *beat weak*. For a mathematical description of these basis functions see Appendix A.1.

4.2.2 Model Description

As described above, NBMs substitute the predictive functions of LBMs, linear models, with FFNNs. These neural networks can be described as a series of (non-linear) transformations of the input data (Bishop, 2006). Using this formalism, we can model the value of each expressive parameter corresponding to score note n_i as the output of a fully-connected FFNN with L *layers* as

$$f_{nbm}(n_i\,;\boldsymbol{\theta}) = \sigma^{(L)}\left(\mathbf{w}^{(L)^{\mathrm{T}}}\boldsymbol{h}^{(L-1)} + w_0^{(L)}\right), \tag{4.1}$$

where $\boldsymbol{h}_i^{(L-1)} \in \mathbb{R}^{D_{L-1}}$ is the activation of the $(L-1)$-th hidden layer (with D_{L-1} units) corresponding to the score note n_i; $\sigma^{(L)}(\cdot)$ is an element-wise activation function and $\mathbf{w}^{(L)} \in \mathbb{R}^{D_{L-1}}$ and $w_0^{(L)} \in \mathbb{R}$ are the vector of weights and a scalar bias of the L-th layer, respectively. The activation for the l-th hidden layer corresponding to the i-th note $\boldsymbol{h}_i^{(l)} \in \mathbb{R}^{D_l}$, where D_l is the number of units in the layer, is given by

$$\boldsymbol{h}_i^{(l)} = \sigma^{(l)}\left(\mathbf{w}^{(l)}\boldsymbol{h}_i^{(l-1)} + \mathbf{w}_0^{(l)}\right), \tag{4.2}$$

where $\mathbf{w}^{(l)} \in \mathbb{R}^{D_l \times D_{l-1}}$ and $\mathbf{w}_0^{(l)} \in \mathbb{R}^{D_l}$ are the matrix of weights[2] and the bias vector of the l-th layer; and $\boldsymbol{h}_i^{(l-1)} \in \mathbb{R}^{D_{l-1}}$ is the activation of the $(l-1)$-th hidden layer. The element-wise activation function of the l-th layer is represented by $\sigma^{(l)}$. As a convention, the 0-th layer represents the input itself, i.e. $\boldsymbol{h}_i^{(0)} = \boldsymbol{\varphi}(n_i)$. The set of all parameters of the network is

[2] Although in this work we follow the convention of denoting vectors with boldface lowercase letters and matrices with boldface uppercase letters, we use $\mathbf{w}$ to denote both the vectors and matrices of weights of the layer of a neural network to highlight that their function is similar to that of the vectors of weights of the linear models described in Chapter 3.

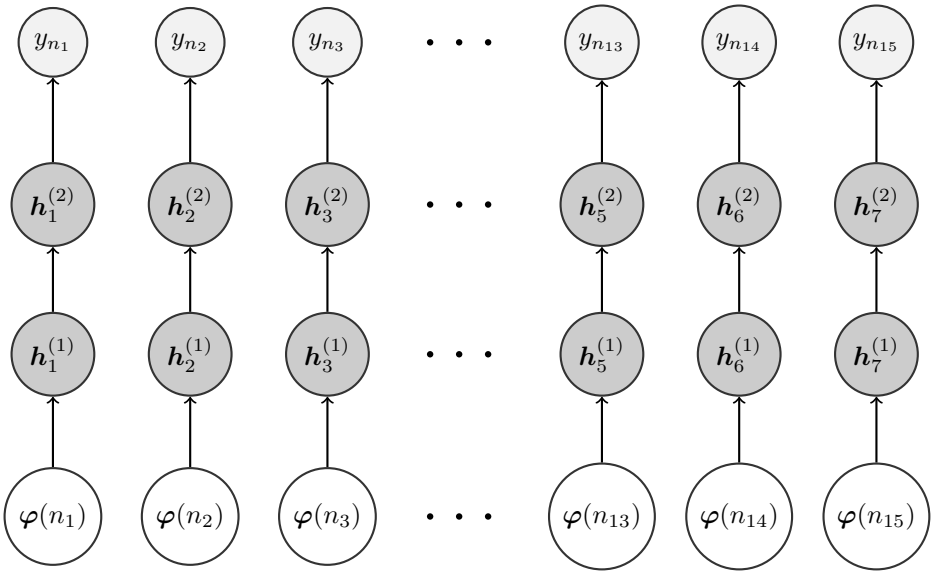


Figure 4.1: Schematic representation of the prediction of an expressive parameter using an NBM consisting of an FFNN with two layers. From bottom to top, the circles represent the input layer, two successive hidden layers and the output layer, respectively; $\boldsymbol{\varphi}(n_i)$ represents the input basis functions for score note n_i, $\boldsymbol{h}_i^{(1)}$ and $\boldsymbol{h}_i^{(2)}$ are the activations of the first and second hidden layers for score note n_i, respectively; and y_{n_i} is the output of the network for score note n_i. The arrows connecting circles represent the flow of information in the FFNN. Note that there are no connections between neighboring notes – the performance of each note is predicted independently of the others, by the same trained FFNN. See Section 4.2.2 for a more detailed explanation.

$\boldsymbol{\theta} = \{\mathbf{w}_0^{(1)}, \mathbf{w}^{(1)}, \ldots, w_0^{(L)}, \mathbf{w}^{(L)}\}$. Common activation functions for the hidden layers are sigmoid, hyperbolic tangent, softmax and rectifier[3]. Since we are using the FFNN in a regression scenario, the activation function of the last layer is set to the identity function ($\sigma^{(L)}(\xi) = \xi$) (Bishop, 2006).

We can then write the complete predictive function for all expressive parameters defined in Section 3.3.1 as

$$F_{nbm}(n_i) = \begin{pmatrix} f_{nbm}^{(\text{vel})}(\boldsymbol{\varphi}(n_i)\,;\boldsymbol{\theta}_{\text{vel}}) \\ f_{nbm}^{(\log \text{bpr})}(\boldsymbol{\varphi}(n_i)\,;\boldsymbol{\theta}_{\log \text{bpr}}) \\ f_{nbm}^{(\text{tim})}(\boldsymbol{\varphi}(n_i)\,;\boldsymbol{\theta}_{\text{tim}}) \\ f_{nbm}^{(\log \text{art})}(\boldsymbol{\varphi}(n_i)\,;\boldsymbol{\theta}_{\log \text{art}}) \end{pmatrix} \tag{4.3}$$

The complete set of parameters of this predictive function is $\boldsymbol{\Theta}_{nbm} = \{\boldsymbol{\theta}_{\text{vel}}, \boldsymbol{\theta}_{\log \text{bpr}}, \boldsymbol{\theta}_{\text{tim}}, \boldsymbol{\theta}_{\log \text{art}}\}$.

Figure 4.1 shows a schematic representation of the prediction of an expressive parameter using an FFNN with two layers.

[3]See Appendix H.3.

Training

As in the case of the LBM, the NBM models expressive parameters independently. Therefore, we can optimize each set of parameters separately. Given a training set $\mathcal{T}$ the parameters of the predictive function can be learned by minimizing the squared error, given by

$$\mathcal{L}_{se}(\boldsymbol{\theta}) = \sum_{i=1}^{N_T} (t_i - f_{nbm}(\boldsymbol{\varphi}(n_i)\,;\boldsymbol{\theta}))^2 \tag{4.4}$$

where t_i is the i-th component of $\mathbf{t}_{\mathcal{T}}$, the vector that concatenates the values of the expressive parameter for all pieces in the training set, and $\boldsymbol{\varphi}(n_i)$ is the i-th row of $\boldsymbol{\Phi}_{\mathcal{T}}$, the matrix concatenating the representation of the scores in the training sets (see Equation (3.22)). Given the nonlinearity of the model, it is not generally possible to compute a closed form analytical solution to $\hat{\boldsymbol{\theta}}_j = \text{argmin}_{\boldsymbol{\theta}_j} \mathcal{L}_{se}(\boldsymbol{\theta}_j)$. Therefore, it is necessary to use numerical methods, such as variants of the *stochastic gradient descent* (SGD) algorithm (Boyd and Vandenberghe, 2004; Nocedal and Wright, 2006; Goodfellow et al., 2016).

In a nutshell, SGD methods update the model parameters iteratively, starting with an initialization value (usually random) and then proceeding to update the value iteratively in the following form:

$$\boldsymbol{\theta}_{q+1} \leftarrow \boldsymbol{\theta}_q - \lambda \frac{\partial}{\partial \boldsymbol{\theta}} \mathcal{L}_{se}(\boldsymbol{\theta}) \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_q} \tag{4.5}$$

i.e., the value of the parameters at the $q+1$-th step is given by the value of the parameters at step q with an update proportional to the gradient of the loss function with respect to the parameters, and λ is a scalar value controlling the size of the step, usually referred to as the *learning rate*. The gradient of the loss function with respect to the model parameters can be efficiently computed using the back-propagation algorithm (Rumelhart et al., 1986). SGD is a first order optimization method, and higher order methods such as Newton's method and Hessian-free optimization might be used (Martens, 2010). In practice, however, variants of SGD that adaptively change the value of the learning rate, such as RMSProp (Tieleman and Hinton, 2012) and Adam (Kingma and Ba, 2014), have proven to be effective, since computing second order updates involves large Hessian matrices, and thus, is computationally expensive.

For most of the experiments reported in this work, we have used RMSProp, a mini-batch variant of SGD where the step size is adaptively updated by a running average of the magnitude of the gradient. RMSProp is controlled by three parameters: the learning rate, a gradient moving average decay factor that controls the decay in the moving average estimation of the magnitude of the gradient, and a small value (i.e. a machine epsilon) for numerical stability. A description of the computation of the parameter updates using RMSProp is provided in Appendix F.1. In order to avoid overfitting, dropout, l_2-norm weight regularization and early stopping are used. Dropout prevents overfitting and provides an efficient way of combining different neural networks by randomly removing units in the network, along with all their incoming and outgoing connections (Srivastava et al., 2014). Regularization of the l_2-norm encourages parameter values to shrink towards zero, unless supported by the data (Bishop, 2006). This regularization is controlled by a scalar value referred to as a regularization coefficient[4]. Early stopping (Morgan and Bourlard, 1990) is performed by monitoring the loss function on a validation set, and stopping the training when there are no more improvements on the loss function on this validation set. In most of our experiments, this validation set comes from splitting the training set into a set for

[4]See Section 3.4.2 and Appendix E.2 for a Bayesian interpretation of regularization of the l_2-norm of the parameters of the model.

computing the parameter updates with 80% of the data, and a validation set with 20% of the data. For FFNNs, weights are randomly initialized as proposed by Glorot and Bengio (2010), and biases are initialized to zero.

The parameters controlling aspects of the training (e.g. learning rate, regularization coefficient and probability of dropout) are referred to as *hyper-parameters*.

4.2.3 Evaluation: Expressive Dynamics

In this section we present a quantitative evaluation of NBMs by comparing the predictive accuracy of LBMs and NBMs predicting expressive dynamics, using the MIDI velocity of performed notes as a proxy for dynamics, as described in Section 3.3.1, by means of 5-fold cross-validation experiments[5].

The main objective of this evaluation is to compare how well the NBM and LBM account for aspects of expressive dynamics. In particular, the cross-validation experiments aim to compare the flexibility of NBMs and LBMs to generalize to new pieces not included in the training set. The second objective of this evaluation is to investigate the effects of the different basis functions describing score information on expressive dynamics predicted by NBMs and LBMs.

For these experiments we used the Magaloff/Chopin and the Zeilinger/Beethoven datasets. The basis function extraction using the groups of basis functions described in Sections 3.3.2 and 4.2.1 on these datasets produced 167 and 163 basis functions, respectively.

Four 5-fold cross-validations were conducted for each model type (LBM and NBM) and dataset (Magaloff/Chopin and Zeilinger/Beethoven) combination. The cross-validations were conducted as follows: We defined training and test sets randomly for each of the 5 partitions (folds) for each dataset, such that each piece in a dataset occurred exactly once per test set. The training set for each partition contained 80% of the pieces in the dataset, and the test set contained the remaining 20%. For each of these partitions, we trained an NBM and LBM on the training set (as described below) and then used the trained models to predict expressive dynamics for each piece in the test set. We evaluated the predictive accuracy of the models by computing the Pearson correlation coefficient r and the coefficient of determination R^2 between the model predictions and the observed dynamics for each piece. We report the average r and R^2 for each model type/dataset combination across all pieces in the dataset.

In the rest of this section, we first provide technical details for training of the LBM and NBM models, followed by the results of the 5-fold cross-validation experiments. Finally, we present a qualitative analysis of the results, in which we use *sensitivity analysis* methods to reveal what relations between the basis functions and the expressive dynamics the LBM and NBM models have learned.

Model Architectures and Training

The LBM models were trained using LSMR, an iterative algorithm for solving sparse LS problems (Fong and Saunders, 2011).

[5] In contrast to the first experiment described in Section 3.5, in this chapter we do not evaluate NBMs in terms of goodness of fit, since FFNNs are known to be universal approximators (Hornik, 1991). This property means that given a large enough architecture, NBMs can potentially reconstruct the training set to an arbitrary level of precision.

Model	Magaloff/Chopin R^2	Magaloff/Chopin r	Zeilinger/Beethoven R^2	Zeilinger/Beethoven r
LBM	0.171	0.470	0.197	0.562
NBM (100, 20)	0.195	0.478	0.266	0.568

Table 4.1: Predictive accuracy for expressive dynamics in terms of the coefficient of determination (R^2) and Pearson correlation coefficient (r), averaged across pieces on each corpora.

The NBMs consist of FFNNs with 2 hidden layers with 100 and 20 rectified linear units, respectively, which we will refer to as NBM (100, 20). The number of layers and units in the NBM (100, 20) architecture were empirically selected from a non-exhaustive search conducted while performing preliminary experiments on joint modeling of expressive dynamics, timing and articulation. This search was performed using 5-fold cross-validations on smaller subsets of the Magaloff/Chopin and Zeilinger/Beethoven datasets (around two thirds of the pieces for each dataset) and a larger subset of basis functions[6].

All NBM models were trained using RMSProp for a maximum of 2000 epochs using a learning rate of 10^{-5}, a gradient moving average decay factor of 0.9 and an epsilon for numerical stability of 10^{-6}, the probability of dropout set to 0.5 and the regularization coefficient equal to 10^{-3}. These hyper-parameters were selected empirically.

Predictive Accuracy

Table 4.1 shows the predictive accuracy of the LBM and the NBM Models in the 5-fold cross-validation scenario on the Magaloff/Chopin and the Zeilinger/Beethoven datasets.

Both measures show a consistent improvement of the NBM model over the LBM model, in particular R^2. This shows that the theoretical benefits of non-linear modeling (non-linear transformations of the input, and interactions between inputs) have practical value in the context of modeling expressive dynamics.

Prior work based on the Magaloff/Chopin data has revealed that a major part of the variance explained by the LBM (captured by R^2) is accounted for by the basis functions that represent dynamic markings and pitch, respectively, whereas other basis functions had very little effect on the predictive accuracy of the model (Grachten and Widmer, 2012). To gain a better insight into the role that different basis functions play in each of the models, the learned models must be studied in more detail. For the LBM this is straightforward: Each of the basis functions is linearly related to the target using a single weight. Therefore, the magnitude of each weight is a direct measure of the impact of its corresponding basis function on the target. In a non-linear model such as the NBM, the weights of the model cannot be interpreted in such a straightforward way. To accommodate for this, we use more generic *sensitivity analysis* methods to investigate the behavior of computational models.

Variance-Based Sensitivity Analysis

In order to investigate the effects of the different basis functions, a *variance based sensitivity analysis* (Saltelli et al., 2010) was performed on the trained LBM and NBM models. In this

[6]The NBM (100, 20) architecture was neither the best for the Magaloff/Chopin dataset nor for the Zeilinger/Beethoven dataset, but performed relatively well given its simplicity.

sensitivity analysis, the model output y is treated as a function of the input basis functions $\boldsymbol{\varphi}$ given the model parameters. The sensitivity of $f(\boldsymbol{\varphi}(n_i)) = y_i$ with respect to the components of the input is explained through a decomposition of its variance into terms depending on the input basis functions and their interactions with each other. The *first order sensitivity coefficient* S_{1_q} measures the individual linear (additive) effect of the q-th basis function φ_q on the model output[7]. On the other hand, the *total effect index* S_{T_q} accounts for the additive effect *plus* all higher order effects of φ_i, including its interactions with the rest of the basis functions. These sensitivity measures are given respectively by

$$S_{1_q} = \frac{\mathrm{var}_{\varphi_i}(\mathbb{E}_{\boldsymbol{\varphi}\setminus\varphi_q}(y \mid \varphi_q))}{\mathrm{var}(y)} \qquad \text{and} \qquad S_{T_q} = \frac{\mathbb{E}_{\boldsymbol{\varphi}\setminus\varphi_q}\left(\mathrm{var}_{\varphi_q}(y \mid \boldsymbol{\varphi}_i)\right)}{\mathrm{var}(y)}, \tag{4.6}$$

where var_{φ_q} is the variance with respect to the q-th basis function, $\mathbb{E}_{\boldsymbol{\varphi}\setminus\varphi_q}$ is the expected value with respect to all basis functions but φ_q, and $\mathrm{var}(y)$ is the total variance of y. From these definitions it is possible to show that $\sum_q S_{1_q} = 1$ and $\sum_q S_{T_q} \geq 1$. Furthermore, it can be shown that for a model whose output depends linearly on its inputs, as is the case with LBMs, both S_{1_q} and S_{T_q} are equal.

Both S_{1_q} and S_{T_q} are estimated using a quasi-Monte Carlo method proposed by Saltelli et al. (2010). This method generates a pseudo random (low-discrepancy) sequence of samples to estimate the expected values and variances in the above equations.

Table 4.2 lists the basis functions that contribute the most to the variance of the model, ordered according to S_1 for the LBM models trained on Magaloff/Chopin and Zeilinger/Beethoven, respectively. The columns labeled *active* specify the percentage of instances where a basis function is non-zero. This is relevant, since a high sensitivity to basis functions that are only very rarely active is a sign of overfitting. For this reason, we have grayed out basis functions that are active in less than 5% of the instances.

Based on the linear model evaluated on the Magaloff/Chopin dataset, it was concluded that pitch and dynamics markings are important factors for predicting expressive dynamics (Grachten and Widmer, 2012). The results reported here, with a more diverse set of basis functions, and evaluated on a second corpus that is independent in terms of both performer and composer, roughly support this conclusion, since both pitch and a few of the most prominent dynamics markings appear as (non-grayed-out) items in the lists. The finding that pitch is (positively) correlated with expressive dynamics is in accordance both with the *High Loud* phrasing rule of the KTH model (Friberg et al., 2006, see Table 1, pp. 148), and with the unsupervised feature learning approach described in (Grachten and Krebs, 2014).

Furthermore, the presence of the *slur incr* and *slur decr* basis functions (see Section 3.3.2) suggests that although the slur marks are only an indication of the ***legato*** articulation, they may act as a proxy for musical grouping, which has been shown to be related to expressive dynamics (Todd, 1992).

Table 4.3 lists the bases to which the NBM model is most sensitive. Roughly speaking, the set of most important bases for the NBM model conveys dynamics markings, pitch, slurs, and duration, as is the case with the LBM model. A notable difference is the high sensitivity of the NBM model to both ***crescendo*** and ***diminuendo*** markings, in both corpora. A plausible explanation for this difference is that although ***crescendo***/***diminuendo*** information is relevant for predicting expressive dynamics, the target cannot be well-approximated as a linear combination of the two basis functions. Comparing the total effect index S_T and the first order sensitivity coefficient S_1 shows that the NBM model has learned interactions involving ***diminuendo*** and ***crescendo***.

[7] To unclutter notation, in this section we write φ_q as a scalar value that can take any possible value in the set of all possible values of $\varphi_q(n)$ for all possible notes n.

Magaloff/Chopin			Zeilinger/Beethoven		
basis function	active	S_1	basis function	active	S_1
pitch	100.00 %	0.168	***sf***	2.98 %	0.209
slur decr	63.06 %	0.067	***smorzando***	0.01 %	0.078
crescendo	42.71 %	0.067	***calando***	0.08 %	0.062
ff	39.12 %	0.059	***crescendo***	26.76 %	0.042
duration	100.00 %	0.035	***ff***	37.48 %	0.036
$\frac{2}{8}$ beat 1	0.01 %	0.034	***f***	26.07 %	0.036
slur incr	62.13 %	0.033	***fp***	0.19 %	0.027
f	35.49 %	0.031	$\frac{5}{4}$ weak	0.01 %	0.026
smorzando	0.01 %	0.025	pitch	100.00 %	0.024
fff	12.86 %	0.023	$\frac{5}{4}$ beat 4	0.00 %	0.020
$\frac{5}{4}$ weak	0.12 %	0.022	***sfp***	0.06 %	0.018
fp	0.01 %	0.021	slur incr	35.92 %	0.018
fz	0.30 %	0.020	duration	100.00 %	0.017
$\frac{3}{8}$ beat 1	0.07 %	0.020	slur decr	37.66 %	0.017
$\frac{12}{8}$ beat 1	0.34 %	0.016	***diminuendo***	18.08 %	0.017

Table 4.2: Basis functions with the largest sensitivity coefficients for the LBM models. Averages are reported over the 5 folds of the cross-validation. Dynamics markings are in bold italic. Basis functions that are non-zero for less than 5% of the instances have been grayed out.

Although these values only indicate that ***diminuendo*** and ***crescendo*** interact with *some* other bases, not necessarily with *each other*, in the following we show that the latter is indeed the case.

Figure 4.2 shows how both the LBM and the NBM behave in two different scenarios concerning the occurrence of a ***crescendo***. For each scenario, a score fragment is shown (taken from the Magaloff/Chopin corpus), that exemplifies the corresponding scenario. The left half of the figure shows the scenario where a ***crescendo*** occurs (indicated by the ramp function in the ***crescendo*** input) without the interference of a ***diminuendo*** (the ***diminuendo*** input is zero). The two graphs below the inputs depict the output of the LBM and NBM, respectively, as a response to these inputs. Apart from a slight non-linearity in the response of the NBM, note that the magnitudes of the responses of both models are virtually equal[8].

In the same way, the right half of the figure shows the response of the models to a ***crescendo*** when preceded by a ***diminuendo***. Note that the basis function encodes the ***diminuendo*** by a ramp from 0 to 1 over the range of the wedge sign, and stays at 1 until the next constant loudness annotation[9], such that over the range of the ***crescendo*** ramp shown in the plot, the ***diminuendo*** basis function is constant at 1.

This is a common situation, as depicted in the musical score fragment, where the musical flow requires a brief (but not sudden) decrease in loudness. Note how the response of the NBM to the ***crescendo*** in this case is much reduced, and also smoother. The response of the LBM, which cannot respond to interactions between inputs, is equal to its response in the first scenario.

[8]The scale of the output is arbitrary, but has been kept constant when plotting the outputs of both the LBM and NBM, to enable visual comparison

[9]This behavior is illustrated for the ***crescendo*** sign in Figure 3.9, and described in detail in Appendix A.1 (see Equation (A.20)).

Magaloff/Chopin				Zeilinger/Beethoven			
basis function	active	S_1	S_T	basis function	active	S_1	S_T
pitch	100.00 %	0.200	0.207	***sf***	2.98 %	0.289	0.300
crescendo	42.71 %	0.037	0.124	***diminuendo***	18.08 %	0.071	0.135
diminuendo	41.56 %	0.036	0.100	duration	100.00 %	0.056	0.126
slur decr	63.06 %	0.044	0.084	***crescendo***	26.76 %	0.046	0.096
ff	39.12 %	0.059	0.083	***f***	26.07 %	0.083	0.095
f	35.49 %	0.051	0.073	slur incr	35.92 %	0.046	0.080
slur incr	62.13 %	0.062	0.073	***ff***	37.48 %	0.052	0.068
duration	100.00 %	0.040	0.072	***p***	64.41 %	0.025	0.034
fff	12.86 %	0.016	0.028	slur decr	37.66 %	0.020	0.033
pp	23.18 %	0.006	0.020	***pp***	40.07 %	0.029	0.032
accent	1.37 %	0.020	0.020	pitch	100.00 %	0.032	0.032
fz	0.30 %	0.018	0.018	***fp***	0.19 %	0.013	0.014
mp	5.46 %	0.007	0.012	ritardando	31.09 %	0.005	0.010
p	41.94 %	0.007	0.012	fermata	0.08 %	0.001	0.004
tot neighbors	2.50 %	0.009	0.011	staccato	8.41 %	0.005	0.005
ppp	5.79 %	0.001	0.011	***sfp***	0.06 %	0.002	0.004

Table 4.3: Basis functions with the largest sensitivity coefficients for the NBM models; Averages are reported over the 5 folds of the cross-validation. Dynamics markings are in bold italic. Basis functions that are non-zero for less than 5% of the instances have been grayed out.

4.2.4 Discussion

The experiments show that the BM models can be used to model expressive dynamics both for different combinations of composers and performers in piano music. A question that has not been explicitly addressed in the experiments is to what degree a model trained on one combination of composer/performer is an accurate model of expressive dynamics in another combination of composer/performer. Although this question is hard to answer in general, it is possible to make some general observations. First of all, along with musical style, performance practice has also evolved over the centuries. For example, a keyboard piece from the Baroque period is typically performed very differently than piano music from the Romantic period. Models trained on one musical style should therefore not be expected to generalize to other styles. Within a specific musical style, expressive styles can still vary substantially from one performer to the other. However, there are substantial commonalities in the expressive dynamics across performers (Repp, 1992, 1994), taking the form of general performance principles. As the sensitivity analysis shows (Section 4.2.3), some of these principles are captured by the model, even if it is trained on the performances of a single performer. This suggests that at least to some extent within a musical style, the models may generalize from one performer to another[10]. However, beyond the search for general performance principles, the BM approach may be used to characterize the individuality of celebrated performers. In Chapter 5 we present preliminary work in this direction.

[10] To test this hypothesis directly, we would need to collect performances of the same pieces/composer by different performers and align them to their scores, which is a very laborious and time-consuming task.

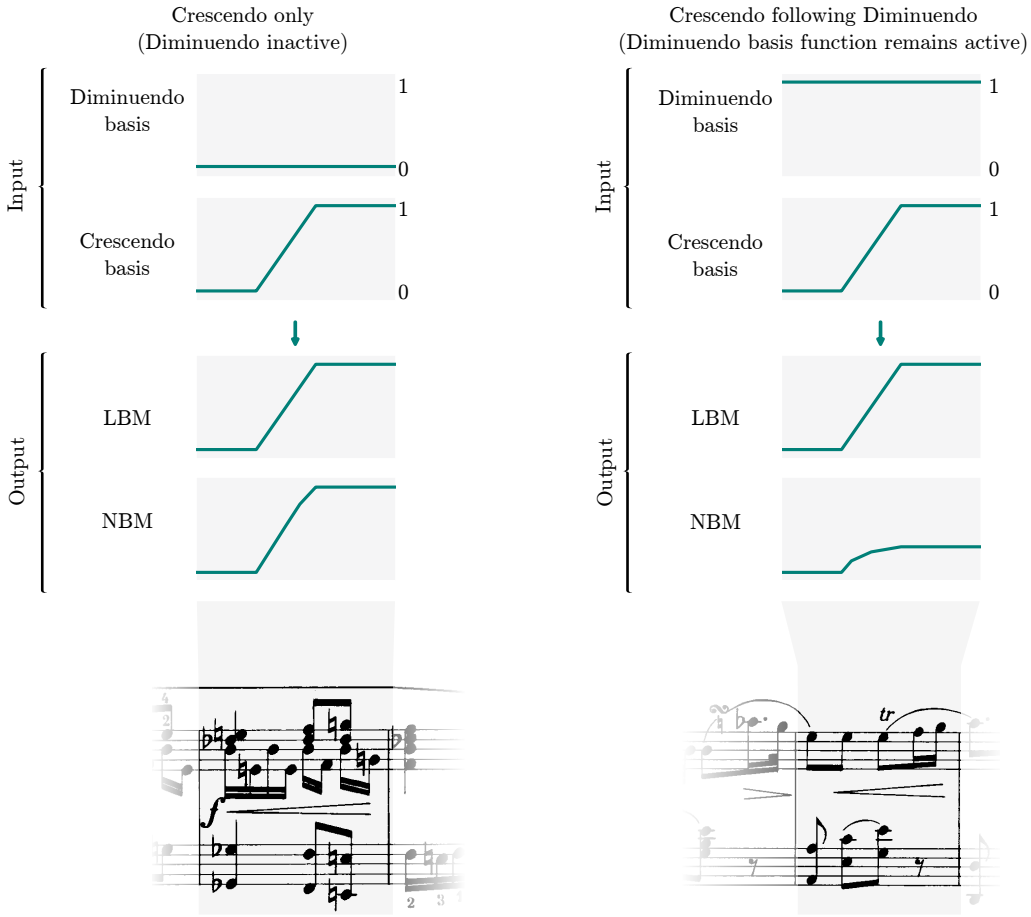


Figure 4.2: Example of the effect of the interaction of *crescendo* after a *diminuendo* for both LBM and NBM.

4.3 Recurrent Non-linear Basis Models

In this section, we introduce the RNBM, an extension of the NBM presented above that uses RNNs to model expressive parameters sequentially. The RNBM was introduced in (Grachten and Cancino-Chacón, 2017).

The rest of this section is structured as follows: Section 4.3.1 discusses the motivation of using sequential models. Section 4.3.2 discusses how to adapt the expressive parameters for sequential models, and introduces a new set of onset-wise and note-wise parameters. Section 4.3.3 presents how to adapt the basis functions for the case of onset-wise and note-wise parameters. Section 4.3.4 presents a mathematical formalization of the RNNs used to model onset-wise parameters. Section 4.3.5 presents a quantitative evaluation of the predictive accuracy for expressive tempo. Finally, Section 4.3.6 discusses these results.

4.3.1 Motivation for Sequential Models

In the FFNN-based NBMs presented in Section 4.2, the relation between the values of the basis functions at note n_i and the value of the expressive target for that note is modeled through intermediate (hidden) layers. These hidden layers are determined only by the current input (i.e. only by the current note), and although FFNNs allow for interactions between current

inputs, they do not allow for interactions between the states of the model at different temporal positions in the piece, or even between other notes occurring at the same temporal position (i.e. other notes in a chord). This behavior is illustrated in Figure 4.1, where there are no arrows indicating the flow of information in the FFNN from a note to another. As such, the FFNN-based NBM is non-sequential: there is no notion of the music as a sequence, or a process.

RNNs are a family of neural networks for modeling sequential data, and have been used successfully for generating text sequences, handwriting synthesis and modeling motion capture data (Graves, 2013). In an RNN, the state of the hidden layer is not only determined by the input at time step i, but also by the hidden state at time step $i-1$ (which in turn is affected by the hidden state at $i-2$, and so forth). As such, the hidden state of an RNN can be regarded as a representation of the prior context of the input – the musical score – up to the current time step. The preparatory function of musical expression (Istók et al., 2013) suggests that when modeling musical expression, not only the prior score context is relevant, but also the upcoming context. In such cases, it can be beneficial to use *bidirectional* RNNs. In this model, one part of the hidden state depends on the prior context, and another part depends on the upcoming context, such that the joint parts form a representation of the temporal context of the current time step in both directions.

4.3.2 Expressive Parameters for Sequential Models: Performance Codec v. 2.0

In this section we define performance codec $\mathcal{C}_{2.0}$, which consists of performance representation model $\mathcal{Y}_{2.0}$ and performance decoder $\mathcal{Y}_{2.0}^{-1}$, which allow for modeling aspects of a performance sequentially.

The expressive parameters defined in performance codec $\mathcal{C}_{1.0}$ in Section 3.3.1 describe the performance of a piano piece in a note-wise fashion: there is a value of the expressive parameters for each note in the score. Nevertheless, adapting these parameters for the case of sequential modeling of polyphonic music is not immediately straightforward. As described above, sequential models such as RNNs process information in a step-wise fashion, and thus, the order in which information is processed by these models is important. On the other hand, the order in which we process information in non-sequential models, such linear models and FFNNs, does not affect the outputs of the model: the predictions will be the same if we present such models with the sequence of score notes (n_0, n_1, n_3) or (n_1, n_3, n_0). The problem of processing information sequentially gets particularly complicated for polyphonic music, since each successive temporal position could have more than one note (i.e. a chord), and thus, there is not a unique answer to which note comes "next".

A solution to this problem could be establishing an ordering of the notes that makes the selection of the next note unambiguous. An example of such ordering would be to take the notes in a chord from lowest to highest (as done in the running example introduced in Section 3.2). While using such an ordering may serve a practical purpose[11], it might not be a good representation of the way humans experience music[12].

In performance representation model $\mathcal{Y}_{2.0}$, we modify the definition of the expressive parameters capturing aspects of expressive dynamics and tempo to have a value for each score onset (i.e. in an onset-wise fashion), and thus, the order of each of these parameters corresponds to the temporal order of the music. Additionally, we define note-wise parameters describing deviations of the individual notes from the local dynamics and tempo. Figure 4.3 presents a visual representation

[11] See for example the model for generating polyphonic music described in (Simon et al., 2017).

[12] From a philosophical standpoint, the experience of music is a very complicated subject. For two contrasting perspectives on this topic see (Kivy, 1991) and (Levinson, 1998).

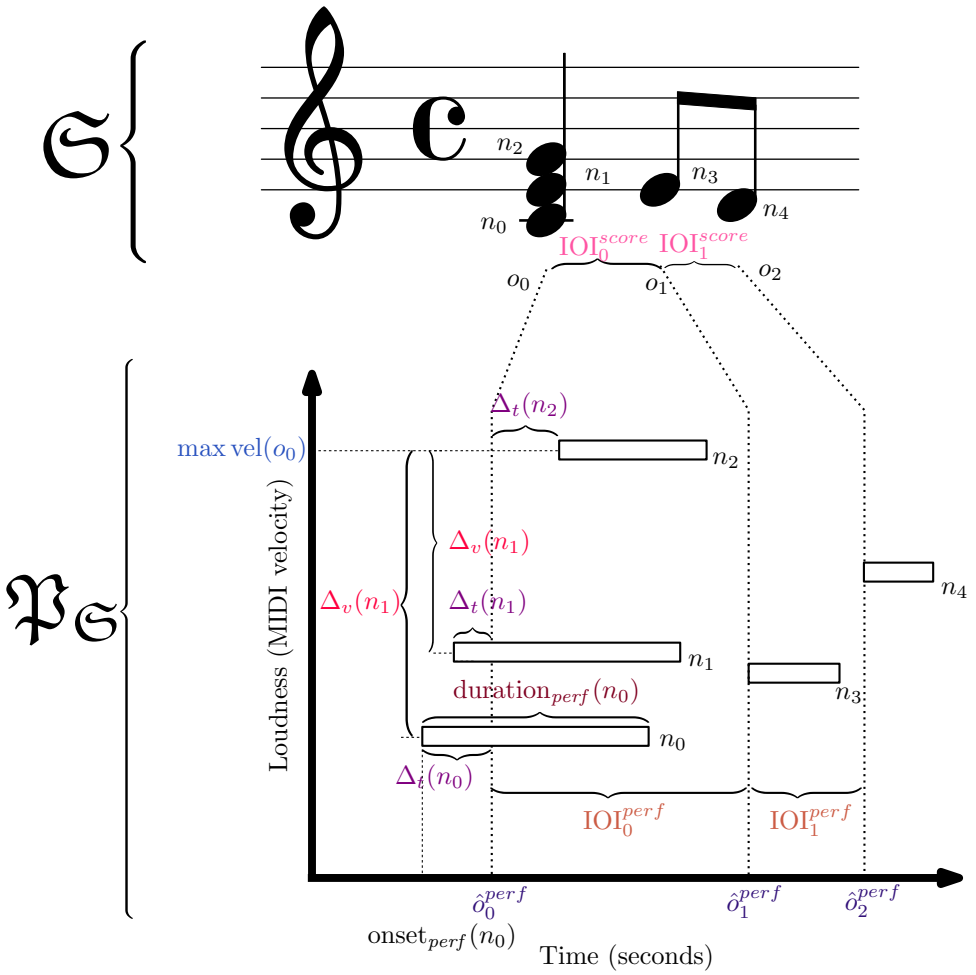


Figure 4.3: Excerpt of a matched MIDI performance $\mathfrak{P}_{\mathfrak{S}}$ (as a piano-roll where the x axis describes time and the y axis describes the MIDI velocity) and its corresponding score $\mathfrak{S}$ showcasing elements for computing the expressive parameters described in performance codec v. 2.0.

of an excerpt of a matched performance in MIDI format which highlights the elements for computing these onset-wise and note-wise parameters.

In the rest of this section, we first define the expressive parameters in performance representation model $\mathcal{Y}_{2.0}$, and then describe the performance decoder $\mathcal{Y}_{2.0}^{-1}$. The definition of the parameters in this performance codec follows the notation conventions established in Section 3.2 and illustrated in Figures 3.2 and 3.4.

Performance Representation Model

The performance representation model $\mathcal{Y}_{2.0}$ defines five expressive parameters divided into two groups: two onset-wise parameters describing temporal aspects of expressive dynamics and tempo, and three note-wise parameters describing deviations in loudness and timing, as well as the articulation of the notes.

These parameters are defined as follows:

Onset-wise Parameters

1. **MIDI velocity trend**. We treat the performed MIDI velocity as a proxy for loudness. This parameter is computed as

$$\mathsf{y}_{\mathrm{vel}_{trend}}(o_i) = \frac{1}{127} \max \mathrm{vel}(o_i), \tag{4.7}$$

where $\mathrm{vel}(o_i) = \{\mathrm{vel}(n_j) \mid n_j \in o_i\}$ represents the performed MIDI velocity of all notes belonging to score onset o_i. See $\max \mathrm{vel}(o_0)$ in the example in Figure 4.3 (in blue). In this example, $\max \mathrm{vel}(o_0)$ is the performed MIDI velocity of score note n_2, which is the maximal performed MIDI velocity of all notes belonging to score onset o_0 (i.e. n_0, n_1 and n_2).

2. **log BPR.** We can naturally adapt the beat period ratio (BPR) defined in Section 3.3.1 to the onset-wise setup by simply computing a value for each score position, rather than a value for each note. This parameter is given by

$$\mathsf{y}_{\log \mathrm{bpr}}(o_i) = \log_2 \frac{\mathrm{BP}(o_i)}{\mathrm{BP}_{ave}}, \tag{4.8}$$

where $\mathrm{BP}(o_i) = \frac{\mathrm{IOI}^{perf}_{o_i}}{\mathrm{IOI}^{score}_{o_i}}$ is the beat period corresponding to score onset o_i and BP_{ave} is the average beat period of the piece (see Equation (3.4)). We associate the values to the start of the IOI, such that the value of BPR at the score onset o_i in the score describes the local tempo between onsets o_i and o_{i+1} (rather than between o_{i-1} and o_i, see column IOI^{score} in Figure 3.4). This is illustrated in Figure 4.3, with IOI^{score}_0 and IOI^{score}_1 (in pink) representing the score IOIs, and IOI^{perf}_0 and IOI^{perf}_1 (in red) representing the performed IOIs corresponding to score onsets o_0 and o_1, respectively.

Note-wise Parameters

3. **MIDI velocity deviations**. We can compute the deviations in loudness for individual notes as the arithmetic difference between the maximal MIDI velocity at a score position, specified by $\mathsf{y}_{\mathrm{vel}_{trend}}$ and the MIDI velocity of the individual note, i.e.

$$\begin{aligned} \mathsf{y}_{\mathrm{vel}_{dev}}(n_i) &= \frac{\Delta_v(n_i)}{127} \\ &= \mathsf{y}_{\mathrm{vel}_{trend}}(o(n_i)) - \frac{\mathrm{vel}(n_j)}{127} \end{aligned} \tag{4.9}$$

These deviations in MIDI velocity are illustrated in Figure 4.3 by $\Delta_v(n_0)$ and $\Delta_v(n_1)$ (in bright red) for score notes n_0 and n_1, respectively.

4. **Timing.** As described in Section 3.3.1, timing refers to the temporal shifting of the performed onset times of each note from the beat grid implied by the local beat period. We compute these timing deviations in the same way as defined in Section 3.3.1, i.e.,

$$\begin{aligned} \mathsf{y}_{\mathrm{tim}}(n_i) &= \Delta_t(n_i) \\ &= \hat{o}^{perf}_i - o^{perf}(n_i) \end{aligned} \tag{4.10}$$

Figure 4.3 illustrates this concept of timing for notes n_0, n_1 and n_2 ($\Delta_t(n_0)$, $\Delta_t(n_1)$ and $\Delta_t(n_2)$ in violet). The above definition of beat period implies that the timing for score onsets consisting of a single note (as is the case of o_1 and o_2 in our running example) is exactly zero, as illustrated in Figure 4.3 for n_3 and n_4 ($\hat{o}^{perf}_1$ and $\hat{o}^{perf}_2$ in indigo).

5. **log Articulation.** As discussed in Section 3.3.1, the articulation of a note refers to the ratio of its performed duration (illustrated in Figure 4.3 in burgundy for n_0) to its notated duration and the current local tempo. We compute articulation in the same way as specified in Section 3.3.1, i.e.,

$$\mathsf{y}_{\log\,\mathrm{art}}(n_i) = \log_2 \frac{\mathrm{duration}_{perf}(n_i)}{\mathrm{duration}(n_i)\mathrm{BP}(o(n_i))}. \tag{4.11}$$

Note that a performance representation under the representation model defined by the above expressive parameters is not simply a matrix, since typically the number of score positions is not equal to the number of notes. In this case, we can write the performance representation as

$$\mathcal{Y}_{2.0}(\mathfrak{S} \mid \mathfrak{P}_{\mathfrak{S}}) = \{\mathbf{T}_{\text{onset-wise}}, \mathbf{T}_{\text{note-wise}}\} \tag{4.12}$$

where $\mathbf{T}_{\text{onset-wise}} \in \mathbb{R}^{N_{N_o} \times 2}$ represents the value of the onset-wise parameters for all score onsets in $\boldsymbol{\mathcal{O}}$, i.e.

$$\mathbf{T}_{\text{onset-wise}} = \begin{pmatrix} \mathsf{y}_{\mathrm{vel}_{trend}}(o_0) & \mathsf{y}_{\log\,\mathrm{bpr}}(o_0) \\ \vdots & \vdots \\ \mathsf{y}_{\mathrm{vel}_{trend}}(o_{N_o}) & \mathsf{y}_{\log\,\mathrm{bpr}}(o_{N_o}) \end{pmatrix} \tag{4.13}$$

and $\mathbf{T}_{\text{note-wise}} \in \mathbb{R}^{N_{N_o} \times 3}$ represents the value of the note-wise parameters for all score notes in $\boldsymbol{\mathcal{X}}$, i.e.

$$\mathbf{T}_{\text{onset-wise}} = \begin{pmatrix} \mathsf{y}_{\mathrm{vel}_{dev}}(n_0) & \mathsf{y}_{\mathrm{tim}}(n_0) & \mathsf{y}_{\log\,\mathrm{art}}(n_0) \\ \vdots & \vdots & \vdots \\ \mathsf{y}_{\mathrm{vel}_{dev}}(n_{N_x}) & \mathsf{y}_{\mathrm{tim}}(n_{N_x}) & \mathsf{y}_{\log\,\mathrm{art}}(n_{N_x}) \end{pmatrix} \tag{4.14}$$

Figure 4.4 shows the full performance representation $\mathcal{Y}_{2.0}(\mathfrak{S} \mid \mathfrak{P}_{\mathfrak{S}}) = \{\mathbf{T}_{\text{note-wise}}, \mathbf{T}_{\text{onset-wise}}\}$ under representation model $\mathcal{Y}_{2.0}$ defined above for the running example introduced in Section 3.2.

Alternatively, we can build a joint performance representation into a single matrix as

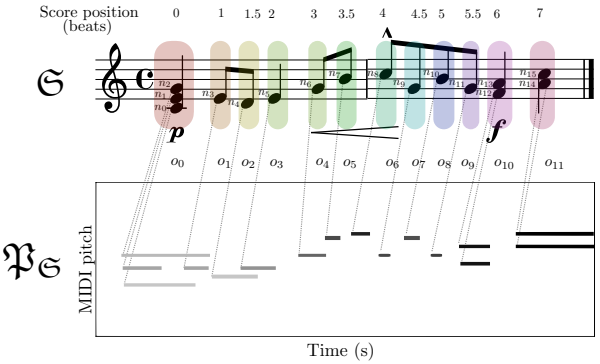
$$\mathbf{T} = \begin{pmatrix} \mathsf{y}_{\mathrm{vel}_{trend}}(o(n_0)) & \mathsf{y}_{\log\,\mathrm{bpr}}(o(n_0)) & \mathsf{y}_{\mathrm{vel}_{dev}}(n_0) & \mathsf{y}_{\mathrm{tim}}(n_0) & \mathsf{y}_{\log\,\mathrm{art}}(n_0) \\ \vdots & \vdots & \vdots & \vdots & \vdots \\ \mathsf{y}_{\mathrm{vel}_{trend}}(o(n_{N_x})) & \mathsf{y}_{\log\,\mathrm{bpr}}(o(n_{N_x})) & \mathsf{y}_{\mathrm{vel}_{dev}}(n_{N_x}) & \mathsf{y}_{\mathrm{tim}}(n_{N_x}) & \mathsf{y}_{\log\,\mathrm{art}}(n_{N_x}) \end{pmatrix}, \tag{4.15}$$

i.e. by duplicating the value of the onset-wise parameters for all notes that belong to the same onset, represented as $o(n_i)$ using the notation conventions from Section 3.2.

Performance Decoder

As in the case of the performance codec defined in Section 3.3.1, this codec is lossless up to the average beat period.

Given a performance representation $\mathbf{T}$ defined in terms of the expressive parameters described above, its score $\mathfrak{S}$, and the average beat period of the performance BP_{ave}, a performance $\mathfrak{P}_{\mathfrak{S}}$ can be reconstructed in MIDI format using performance decoder $\mathcal{Y}_{2.0}^{-1}$.



$\mathcal{Y}_{2.0}$

$\mathbf{T}_{\text{note-wise}}$

score pos.	Note	$y_{\text{vel}_{dev}}(\cdot)$	$y_{\text{tim}}(\cdot)$	$y_{\text{log art}}(\cdot)$
0.0	n_0	0.126	-0.014	0.230
0.0	n_1	0.000	0.000	-0.624
0.0	n_2	0.079	0.014	0.551
1.0	n_3	0.000	0.000	-0.152
1.5	n_4	0.000	0.000	0.697
2.0	n_5	0.000	0.000	-0.690
3.0	n_6	0.000	0.000	0.113
3.5	n_7	0.000	0.000	-0.619
4.0	n_8	0.000	0.000	-0.534
4.5	n_9	0.000	0.000	-1.037
5.0	n_{10}	0.000	0.000	-0.675
5.5	n_{11}	0.000	0.000	-1.379
6.0	n_{12}	0.016	-0.002	-0.889
6.0	n_{13}	0.000	0.002	-0.841
7.0	n_{14}	0.008	-0.002	0.023
7.0	n_{15}	0.000	0.002	0.000

$\mathbf{T}_{\text{onset-wise}}$

score pos.	Onset	$y_{\text{vel}_{trend}}(\cdot)$	$y_{\text{log bpr}}(\cdot)$
0.0	o_0	0.346	0.091
1.0	o_1	0.346	-0.061
1.5	o_2	0.220	0.016
2.0	o_3	0.417	0.013
3.0	o_4	0.591	-0.142
3.5	o_5	0.693	-0.114
4.0	o_6	0.850	-0.041
4.5	o_7	0.709	-0.206
5.0	o_8	0.677	-0.101
5.5	o_9	0.732	0.028
6.0	o_{10}	0.913	-0.053
7.0	o_{11}	0.976	0.455

Figure 4.4: Performance representation $\mathcal{Y}_{2.0}(\mathfrak{S} \mid \mathfrak{P}_{\mathfrak{S}}) = \{\mathbf{T}_{\text{note-wise}}, \mathbf{T}_{\text{onset-wise}}\}$ for matched performance $\mathfrak{P}_{\mathfrak{S}}$ (with score $\mathfrak{S}$) for the running example introduced in Section 3.2 (see Figure 3.2). Colored rectangles highlighting regions of $\mathbf{T}_{\text{note-wise}}$ and $\mathbf{T}_{\text{onset-wise}}$ correspond to the score position highlighted with the same color.

Algorithm 4.1 presents the performance decoder $\mathcal{Y}_{2.0}^{-1}$. The main difference between performance decoders $\mathcal{Y}_{1.0}^{-1}$ and $\mathcal{Y}_{2.0}^{-1}$ is that in the latter, a performance representation consists of two matrices describing the onset- and note-wise parameters, respectively. In order to compute the MIDI velocity of each note, we have to first compute the maximal velocity per onset, and then we compute the deviations of MIDI velocity for the individual notes.

4.3.3 Adapting Basis Functions for Onset-wise and Note-wise Parameters

Although the basis functions defined in Sections 3.3.2, 3.4.1 and 4.2.1 described score information at the note level (i.e. note-wise), it is easy to see that some of the groups of basis functions are more naturally defined at the score onset level. Examples of these functions include the groups of basis functions describing dynamics, as seen in Figure 3.9 for ***p***, ***f*** and ***crescendo***. On the other hand, some basis functions like those in the polynomial pitch model, or those encoding accents, are inherently defined for each note, as seen in Figure 3.9 for ***marcato*** and pitch.

Algorithm 4.1: Performance Decoder v. 2.0

Input:

- $\mathfrak{S}$: Score
  - $\boldsymbol{\mathcal{X}}$: Set of notes in score $\mathfrak{S}$ with cardinality N_x
  - $\boldsymbol{\mathcal{O}}$: Set of onsets in score $\mathfrak{S}$ with cardinality N_o (we assume that the onsets are ordered by score position)
- $\mathbf{T} = \{\mathbf{T}_{\text{onset-wise}}, \mathbf{T}_{\text{note-wise}}\}$: Performance representation (see Equation (4.12))
- BP_{ave}: Average beat period

Output: Information to reconstruct MIDI file:

- `Pitch`: MIDI pitch of each note in the score. Initialized as $\texttt{Pitch} = \mathbf{0} \in \mathbb{Z}^{N_x}$.
- `Velocity`: Performed MIDI velocity of each note in the score. Initialized as $\texttt{Velocity} = \mathbf{0} \in \mathbb{Z}^{N_x}$.
- `Onset`: Performed onset time in seconds of each note in the score. Initialized as $\texttt{Onset} = \mathbf{0} \in \mathbb{R}^{N_x}$.
- `Duration`: Performed duration in seconds of each note in the score. Initialized as $\texttt{Duration} = \mathbf{0} \in \mathbb{R}^{N_x}$.

for $0 \leq i \leq N_o - 1$ **do**

Compute the maximal MIDI velocity for score onset o_i as

$$\text{vel}_{max}(o_i) = 127 \times t_{i;\text{vel}_{trend}} \tag{4.16}$$

where $t_{i;\text{vel}_{trend}}$ is the component in the i-th row of $\mathbf{T}_{\text{onset-wise}}$ corresponding to the MIDI velocity trend parameter defined in Equation (4.7).

Compute the beat period for score onset o_i as

$$\text{BP}(o_i) = \text{BP}_{ave} \times 2^{t_{i;\log \text{bpr}}} \tag{4.17}$$

where $t_{i;\log \text{bpr}}$ is the component in the i-th row of $\mathbf{T}_{\text{onset-wise}}$ corresponding to the log BPR defined in Equation (4.8).

if $i = 0$ **then**

Initialize the first equivalent onset

$$\hat{o}_0^{perf} = 0 \tag{4.18}$$

else

Compute the equivalent onset

$$\hat{o}_i^{perf} = \hat{o}_{i-1}^{perf} + \text{BP}(o_i) \times \text{IOI}_i^{score}, \tag{4.19}$$

where IOI_i^{score} is the score onset corresponding to onset o_i, as defined in Equation (3.5) (see Section 3.3.1).

for $0 \leq i \leq N_x - 1$ **do**

Get MIDI pitch of score note n_i

$$\texttt{Pitch}_i = \text{pitch}(n_i) \tag{4.20}$$

Compute MIDI velocity of score note n_i

$$\texttt{Velocity}_i = \text{vel}_{max}(o(n_i)) - 127 \times t_{i;\text{vel}_{dev}}, \tag{4.21}$$

where $t_{i;\text{vel}_{dev}}$ is the component in the i-th row of $\mathbf{T}_{\text{note-wise}}$ corresponding to the MIDI velocity deviations parameter defined in Equation (4.9), and $\text{vel}_{max}(o(n_i))$ is the maximal MIDI velocity corresponding to the score onset to which n_i belongs.

Compute the onset time for score note n_i

$$\texttt{Onset}_i = \hat{o}^{perf}(n_i) - t_{i;\text{tim}} \tag{4.22}$$

where $t_{i;\text{tim}}$ is the component in the i-th row of $\mathbf{T}_{\text{note-wise}}$ corresponding to the timing parameter defined in Equation (4.10), and $\hat{o}^{perf}(n_i)$ is the equivalent performed onset corresponding to the score onset to which n_i belongs.

Compute duration for score note n_i

$$\texttt{Duration}_i = 2^{t_{i;\log \text{art}}} \times \text{duration}(n_i) \times \text{BP}(o(n_i)), \tag{4.23}$$

where $t_{i;\log \text{art}}$ is the component in the i-th row of $\mathbf{T}_{\text{note-wise}}$ corresponding to the log articulation ratio defined in Equation (4.11) and $\text{BP}(o(n_i))$ is the beat period corresponding to the score onset to which n_i belongs.

For models defined for onset-wise parameters, the value of the j-th basis function at score position o_i is given as the average of the value of the basis functions corresponding to the notes at that score position, i.e.

$$\varphi_j(o_i) = \frac{1}{|o_i|} \sum_{n_q \in o_i} \varphi_j(n_q). \tag{4.24}$$

In the case of models defined for onset-wise and note-wise parameters, the representation of a score $\mathfrak{S}$ can be written as

$$\mathcal{F}(\mathfrak{S}) = \{\mathbf{\Phi}_{\text{onset-wise}}, \mathbf{\Phi}_{\text{note-wise}}\} \tag{4.25}$$

where $\mathbf{\Phi}_{\text{onset-wise}} \in \mathbb{R}^{N_{N_o} \times M_{\text{ow}}}$ and $\mathbf{\Phi}_{\text{note-wise}} \in \mathbb{R}^{N_{N_x} \times M_{\text{nw}}}$ can be written as

$$\mathbf{\Phi}_{\text{onset-wise}} = \begin{pmatrix} \varphi_1(o_0) & \dots & \varphi_{M_{\text{ow}}}(o_0) \\ \vdots & \ddots & \vdots \\ \varphi_1(o_{N_o-1}) & \dots & \varphi_{M_{\text{ow}}}(o_{N_o-1}) \end{pmatrix} \tag{4.26}$$

$$\mathbf{\Phi}_{\text{note-wise}} = \begin{pmatrix} \varphi_1(n_0) & \dots & \varphi_{M_{\text{nw}}}(n_0) \\ \vdots & \ddots & \vdots \\ \varphi_1(n_{N_x-1}) & \dots & \varphi_{M_{\text{nw}}}(n_{N_x-1}). \end{pmatrix}. \tag{4.27}$$

Figure 4.5 shows the score representation $\mathcal{F}(\mathfrak{S}) = \{\mathbf{\Phi}_{\text{onset-wise}}, \mathbf{\Phi}_{\text{note-wise}}\}$ of the running example introduced in Section 3.2 for a performance representation model consisting of basis functions describing pitch, dynamics markings ***p***, ***f*** and ***crescendo***; and a basis function encoding an accent (a ***marcato***).

It is important to note that the averaging of basis functions per onset is an implementation necessity, and could result in unmusical representations of the score. For example, in some contexts the average pitch of an onset might not be musically meaningful. Therefore, in certain cases it might be better to only use the subset of meaningful basis functions to model an expressive parameter. More basis functions that better capture the sequential nature of music will be discussed in Chapters 6.

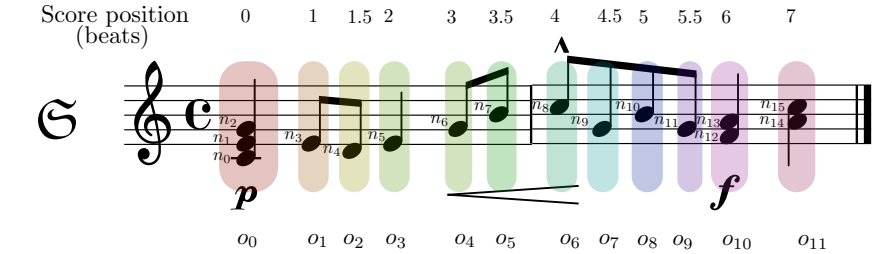
4.3.4 Model Description

In contrast to the LBMs and NBMs described in Chapter 3 and Section 4.2, respectively, the complete predictive function of an RNBM model consists of *two components*: a group of sequential functions for predicting the onset-wise parameters (MIDI velocity deviations, timing and log articulation), and a group of non-sequential functions for predicting the note-wise parameters (MIDI velocity trend and log BPR).

We use FFNNs as predictive functions for the note-wise parameters, and RNNs as predictive functions for the onset-wise parameters. Since a detailed description of FFNNs was already presented in Section 4.2.2, in this section, we discuss how RNNs are used to model expressive parameters.

We can model the value of an expressive parameter corresponding to the score onset o_i as the output of an RNN with L hidden layers as

$$f_{rnbm}(n_i\,; \boldsymbol{\theta}) = \sigma^{(L)}\left(\mathbf{w}^{(L)^{\mathrm{T}}} \boldsymbol{h}_i^{(L-1)} + w_0^{(L)}\right), \tag{4.28}$$



$\mathcal{F}$

$\boldsymbol{\Phi}_{\text{note-wise}}$

score pos.	Note	φ_{pitch}	$\varphi_{\boldsymbol{p}}$	$\varphi_{\boldsymbol{f}}$	$\varphi_{\boldsymbol{cresc}}$	φ_{accent}
0.0	n_0	$\frac{60}{127}$	1.0	0.0	0.0	0.0
0.0	n_1	$\frac{64}{127}$	1.0	0.0	0.0	0.0
0.0	n_2	$\frac{67}{127}$	1.0	0.0	0.0	0.0
1.0	n_3	$\frac{64}{127}$	1.0	0.0	0.0	0.0
1.5	n_4	$\frac{62}{127}$	1.0	0.0	0.0	0.0
2.0	n_5	$\frac{64}{127}$	1.0	0.0	0.0	0.0
3.0	n_6	$\frac{67}{127}$	1.0	0.0	0.3	0.0
3.5	n_7	$\frac{71}{127}$	1.0	0.0	0.7	0.0
4.0	n_8	$\frac{72}{127}$	1.0	0.0	1.0	1.0
4.5	n_9	$\frac{67}{127}$	1.0	0.0	1.0	0.0
5.0	n_{10}	$\frac{71}{127}$	1.0	0.0	1.0	0.0
5.5	n_{11}	$\frac{67}{127}$	1.0	0.0	1.0	0.0
6.0	n_{12}	$\frac{65}{127}$	0.0	1.0	0.0	0.0
6.0	n_{13}	$\frac{69}{127}$	0.0	1.0	0.0	0.0
7.0	n_{14}	$\frac{69}{127}$	0.0	1.0	0.0	0.0
7.0	n_{15}	$\frac{72}{127}$	0.0	1.0	0.0	0.0

averaging

$\boldsymbol{\Phi}_{\text{onset-wise}}$

score pos.	Onset	φ_{pitch}	$\varphi_{\boldsymbol{p}}$	$\varphi_{\boldsymbol{f}}$	$\varphi_{\boldsymbol{cresc}}$	φ_{accent}
0.0	o_0	$\frac{63.7}{127}$	1.0	0.0	0.0	0.0
1.0	o_1	$\frac{64.0}{127}$	1.0	0.0	0.0	0.0
1.5	o_2	$\frac{62.0}{127}$	1.0	0.0	0.0	0.0
2.0	o_3	$\frac{64.0}{127}$	1.0	0.0	0.0	0.0
3.0	o_4	$\frac{67.0}{127}$	1.0	0.0	0.3	0.0
3.5	o_5	$\frac{71.0}{127}$	1.0	0.0	0.7	0.0
4.0	o_6	$\frac{72.0}{127}$	1.0	0.0	1.0	1.0
4.5	o_7	$\frac{67.0}{127}$	1.0	0.0	1.0	0.0
5.0	o_8	$\frac{71.0}{127}$	1.0	0.0	1.0	0.0
5.5	o_9	$\frac{67.0}{127}$	1.0	0.0	1.0	0.0
6.0	o_{10}	$\frac{67.0}{127}$	0.0	1.0	0.0	0.0
7.0	o_{11}	$\frac{70.5}{127}$	0.0	1.0	0.0	0.0

Figure 4.5: Note-wise ($\boldsymbol{\Phi}_{\text{note-wise}}$) and onset-wise ($\boldsymbol{\Phi}_{\text{onset-wise}}$) components of the score representation of $\mathcal{S}$ under score representation model $\mathcal{F} = \{\varphi_{\boldsymbol{p}}, \varphi_{\boldsymbol{f}}, \varphi_{\boldsymbol{cresc}}, \varphi_{\text{accent}}\}$. Colored rectangles highlighting regions of $\boldsymbol{\Phi}_{\text{note-wise}}$ and $\boldsymbol{\Phi}_{\text{onset-wise}}$ correspond to the score position highlighted with the same color.

where the size of the hidden layers, element-wise activation functions, weights and biases are equivalent to those described in Section 4.2.2. The main difference between fully-connected FFNNs and RNNs is that the latter allow for using both recurrent and fully connected hidden layers. The output of a recurrent layer for score onset o_i can be written as

$$\boldsymbol{h}_i^{(l)} = \sigma^{(l)}\left(g_{\varphi}^{(l)}\left(\boldsymbol{\varphi}(o_i)\right) + g_h^{(l)}\left(\boldsymbol{h}_{i^*}\right)\right), \tag{4.29}$$

where $g_{\varphi}\left(\boldsymbol{\varphi}(o_i)\right)$ represents the contribution of the input of the network at time i (i.e. score onset o_i), $g_h\left(\boldsymbol{h}_{i^*}\right)$ is the contribution of other time steps (past or future, or a combination of both) of the state of the recurrent layer. As in the case of NBMs, $\sigma^{(l)}(\cdot)$ is an element-wise activation function. A vanilla recurrent layer is given as

$$\boldsymbol{h}_i^{(l)} = \sigma^{(l)}\left(\mathbf{w}_x^{(l)}\boldsymbol{h}_i^{(l-1)} + \mathbf{w}_h^{(l)}\boldsymbol{h}_{i-1}^{(l)} + \mathbf{w}_0^{(l)}\right). \tag{4.30}$$

where $\mathbf{w}_x^{(l)} \in \mathbb{R}^{D_l \times D_{l-1}}$ is a matrix of weights connecting the $(l-1)$-th layer $\boldsymbol{h}_i^{(l)}$, $\mathbf{w}_h^{(l)} \in \mathbb{R}^{D_l \times D_{l-1}}$ is a matrix of weights connecting the $(i-1)$-th activation of $\boldsymbol{h}^{(l)}$; and $\mathbf{w}_0^{(l)} \in \mathbb{R}^{D_l}$ are the matrix of weights and the bias vector of the l-th layer. While it is theoretically possible for large enough vanilla RNNs to predict sequences of arbitrary complexity, it has been shown

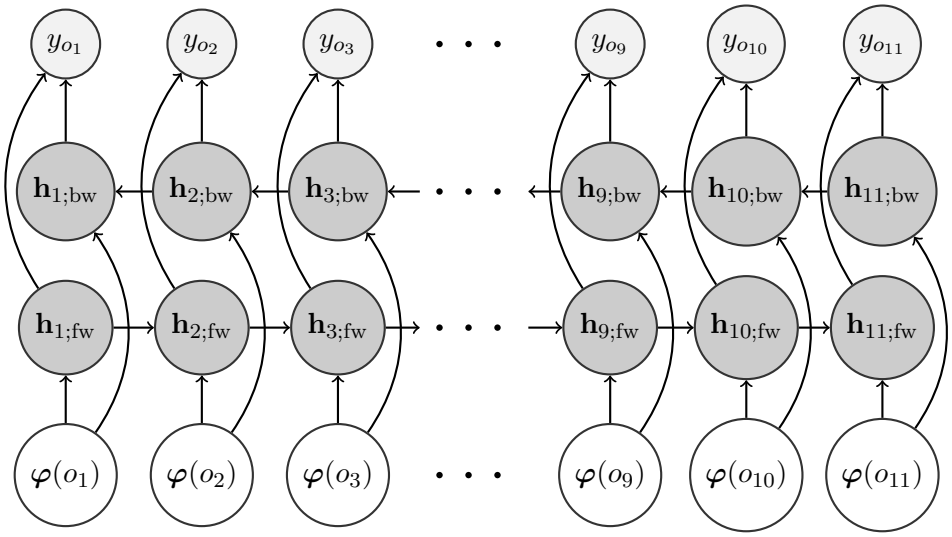


Figure 4.6: Schematic representation of the prediction of an expressive parameter using an RNBM consisting of an RNN with a single bidirectional layer. From bottom to top, the circles represent the input layer, the forward and backward components of the bidirectional layer, and the output layer, respectively ($\boldsymbol{\varphi}(o_i)$ represents the input basis functions for score onset o_i, $\boldsymbol{h}_{i;\text{fw}}$ and $\boldsymbol{h}_{i;\text{bw}}$ are the activations of the forward and backward components of the recurrent hidden layer for score onset o_i, respectively; and y_{o_i} is the output of the network for score note o_i). From left to right, advancing time steps are shown. The arrows connecting circles represent the flow of information in the RNN. See Section 4.3.4 for a more detailed explanation.

that numerical limitations of training algorithms do not allow them to properly model long temporal dependencies (Pascanu et al., 2013). To address these problems, the Long Short Term Memory (LSTM) architecture was proposed in (Hochreiter and Schmidhuber, 1997). LSTMs include special purpose recurrent layers with memory cells that allow for exploiting long range dependencies in the data. Appendix F.2 provides a more mathematical description of recurrent layers used in this work.

As discussed in Section 4.3.1, we use bidirectional RNNs (Schuster and Paliwal, 1997) for modeling expressive parameters. In this way, we combine score information from the past and future to make a prediction of the expressive performance of the current score position. The main components of such a network are bidirectional layers, which are composite layers consisting of two recurrent layers, a forward recurrent layer $\boldsymbol{h}^{(l)}_{i;\text{fw}}$, which processes information from the beginning to the end of an input sequence; and a backward recurrent layer $\boldsymbol{h}^{(l)}_{i;\text{bw}}$, which processes information from the end to the start of a sequence. Common ways to build these composite layers are by concatenating the forward and backward layers into a single vector, or making a linear combination of the output of the layers, i.e.,

$$\boldsymbol{h}^{(l)}_{i;bd} = \begin{pmatrix} \boldsymbol{h}^{(l)}_{i;\text{fw}} \\ \boldsymbol{h}^{(l)}_{i;\text{bw}} \end{pmatrix}, \qquad \text{or} \qquad \boldsymbol{h}^{(l)}_{i;bd} = \alpha_f \boldsymbol{h}^{(l)}_{i;\text{fw}} + \alpha_b \boldsymbol{h}^{(l)}_{i;\text{bw}}, \tag{4.31}$$

where $\alpha_f, \alpha_b \in \mathbb{R}$ are weighting coefficients. The structure of a bidirectional RNN for modeling an expressive parameter is illustrated in Figure 4.6. Note that bidirectional RNNs are not causal models.

We can write the complete predictive function for all expressive parameters defined in Section

3.3.1 for score $\mathfrak{S}$ as

$$F_{rnbm}(\boldsymbol{\Phi}, \boldsymbol{\theta}) = \{\mathbf{Y}_{\text{onset-wise}}, \mathbf{Y}_{\text{note-wise}}\}, \tag{4.32}$$

where $\mathbf{Y}_{\text{onset-wise}} \in \mathbb{R}^{N_{N_o} \times 2}$ represents the predictions of the onset-wise parameters for all score onsets in $\boldsymbol{\mathcal{O}}$ using RNNs, i.e.

$$\mathbf{Y}_{\text{onset-wise}} = \begin{pmatrix} f_{rnbm}^{(\text{vel}_{trend})}(\boldsymbol{\varphi}(o_0)\,;\boldsymbol{\theta}_{\text{vel}_{trend}}) & f_{rnbm}^{(\text{bpr})}(\boldsymbol{\varphi}(o_0)\,;\boldsymbol{\theta}_{\text{bpr}}) \\ \vdots & \vdots \\ f_{rnbm}^{(\text{vel}_{trend})}(\boldsymbol{\varphi}(o_{N_o-1})\,;\boldsymbol{\theta}_{\text{vel}_{trend}}) & f_{rnbm}^{(\text{bpr})}(\boldsymbol{\varphi}(o_{N_o-1})\,;\boldsymbol{\theta}_{\text{bpr}}) \end{pmatrix}, \tag{4.33}$$

where the subscript *rnbm* denotes that the predictive function is an RNN; and $\mathbf{Y}_{\text{note-wise}} \in \mathbb{R}^{N_{N_o} \times 3}$ represents the predictions of note-wise parameters for all score notes in $\boldsymbol{\mathcal{X}}$ computed using FFNNs, i.e.

$$\mathbf{Y}_{\text{onset-wise}} = \begin{pmatrix} f_{nbm}^{(\text{vel}_{dev})}(\boldsymbol{\varphi}(n_0)\,;\boldsymbol{\theta}_{\text{vel}_{dev}}) & f_{nbm}^{(\text{tim})}(\boldsymbol{\varphi}(n_0)\,;\boldsymbol{\theta}_{\text{tim}}) & f_{nbm}^{(\text{art})}(\boldsymbol{\varphi}(n_0)\,;\boldsymbol{\theta}_{\text{art}}) \\ \vdots & \vdots & \vdots \\ f_{nbm}^{(\text{vel}_{dev})}(\boldsymbol{\varphi}(n_{N_x-1})\,;\boldsymbol{\theta}_{\text{vel}_{dev}}) & f_{nbm}^{(\text{tim})}(\boldsymbol{\varphi}(n_{N_x-1})\,;\boldsymbol{\theta}_{\text{tim}}) & f_{nbm}^{(\text{art})}(\boldsymbol{\varphi}(n_{N_x-1})\,;\boldsymbol{\theta}_{\text{art}}), \end{pmatrix}, \tag{4.34}$$

where the subscript *nbm* denotes that the predictive function is an FFNN.

In a similar fashion as the performance representation described in Equation (4.15), we can write the output of the predictive function for score $\mathfrak{S}$ as a single matrix where each row represents the predictions of the expressive parameters for each note in the score as

$$\mathbf{Y} = \begin{pmatrix} f_{rnbm}^{(\text{vel}_{trend})}(\boldsymbol{\varphi}(o(n_0))\,;\boldsymbol{\theta}_{\text{vel}_{trend}}) & \dots & f_{rnbm}^{(\text{vel}_{trend})}(\boldsymbol{\varphi}(o(n_{N_x-1}))\,;\boldsymbol{\theta}_{\text{vel}_{trend}}) \\ f_{rnbm}^{(\text{bpr})}(\boldsymbol{\varphi}(o(n_0))\,;\boldsymbol{\theta}_{\text{bpr}}) & \dots & f_{rnbm}^{(\text{bpr})}(\boldsymbol{\varphi}(o(n_{N_x-1}))\,;\boldsymbol{\theta}_{\text{bpr}}) \\ f_{nbm}^{(\text{vel}_{dev})}(\boldsymbol{\varphi}(n_0)\,;\boldsymbol{\theta}_{\text{vel}_{dev}}) & \dots & f_{nbm}^{(\text{vel}_{dev})}(\boldsymbol{\varphi}(n_{N_x-1})\,;\boldsymbol{\theta}_{\text{vel}_{dev}}) \\ f_{nbm}^{(\text{tim})}(\boldsymbol{\varphi}(n_0)\,;\boldsymbol{\theta}_{\text{tim}}) & \dots & f_{nbm}^{(\text{tim})}(\boldsymbol{\varphi}(n_{N_x-1})\,;\boldsymbol{\theta}_{\text{tim}}) \\ f_{nbm}^{(\text{art})}(\boldsymbol{\varphi}(n_0)\,;\boldsymbol{\theta}_{\text{art}}) & \dots & f_{nbm}^{(\text{art})}(\boldsymbol{\varphi}(n_{N_x-1})\,;\boldsymbol{\theta}_{\text{art}}) \end{pmatrix}^{\mathrm{T}}, \tag{4.35}$$

where the values of the predicted parameters for the notes belonging to the same onset are duplicated for onset-wise models[13].

Training

As in the case of LBMs and NBMs, RNBMs model expressive parameters independently, and thus we can optimize each set of parameters separately. The training of the FFNNs modeling note-wise parameters is performed in the same fashion as described in Section 4.2.2. RNNs modeling onset-wise parameters can be trained very similarly by minimizing the squared error using (a variant of) SGD, where the gradient of the loss function with respect to the model parameters can be efficiently computed using the *back-propagation through time* (BPTT) algorithm (Rumelhart et al., 1986).

As with FFNNs, we used RMSProp for training RNNs. We found it more effective to initialize the weights of the recurrent layers drawing from uniform distributions. The biases and initial states of the network are initialized to zero. Since the value of the gradients can sometimes become excessively large, to avoid numerical problems the value of the gradients is clipped to lie within a predefined range, and the number of time steps considered in the back-propagated gradient is truncated (Graves, 2013).

For a more elaborate review of RNNs see (Graves, 2013) and (Goodfellow et al., 2016).

[13] This matrix is transposed for space constraints.

4.3.5 Experiments: Expressive Tempo

In this section we present a quantitative evaluation of the sequential components of RNBMs by comparing the predictive accuracy of linear models, FFNNs, and RNNs for expressive tempo, using the log BPR parameter described in Section 4.3.2, by means of 10-fold cross-validation experiments. As mentioned for the case of the FFNNs, RNNs are also known to be universal approximators (Schäfer and Zimmermann, 2006). Therefore, in this chapter we do not evaluate RNNs in terms of goodness of fit, since given a large enough neural architecture, RNNs are able to fit the training data to an arbitrary level of precision.

The objective of this evaluation is to compare how well the RNBMs, NBMs and LBMs account for aspects of expressive tempo. In particular, the cross-validation experiments aim to compare the flexibility of these models to generalize to new pieces not included in the training set. A second objective of this evaluation is to analyze the contribution of past and future score information on the performance of a note.

Three 10-fold cross-validations were conducted for each model type (LBM, NBM and RNBM) using the Magaloff/Chopin dataset. The basis function extraction using the groups of basis functions described in Sections 3.3.2 and 4.2.1, adapted for the sequential setup as described in Section 4.3.3, generated 220 basis functions. The cross-validations were conducted as follows: We defined training and test sets randomly for each of the 10 folds, such that each piece in the dataset occurred exactly once per test set. The training set for each fold contained 80% of the pieces (on average 139.5) in the dataset, and the test set contained the remaining 20% (15.5 on average). For each of these folds, we trained an RNBM, NBM and LBM on the training set (as described below) and then used the trained models to predict expressive tempo for each piece in the test set. The predictive accuracy of the models was evaluated by computing the Pearson correlation coefficient r and the coefficient of determination R^2 between the model predictions and the performed tempo for each piece. We report the average r and R^2 for each model across all pieces in the dataset.

Model Architectures and Training

All LBMs were trained using LSMR (Fong and Saunders, 2011).

The FFNN architecture used in this evaluation consists of a single hidden layer with 20 rectified linear units, and a linear output layer with a single unit. We will refer to this architecture as NBM (20). The architecture of the RNNs consists of a single bidirectional vanilla recurrent layer with 20 rectified linear units and a linear output layer with a single unit. The bidirectional recurrent layer of this architecture uses a linear combination of the outputs of the forward and backward layers, as described in Equation (4.31), with $\alpha_f = \alpha_b = 1$. We refer to this architecture as RNBM (20).

Further hyper-parameters were set as follows. The weight parameters of the FFNNs were initialized using the method proposed by Glorot and Bengio (2010). For the RNN models, initializing the weights of recurrent layers simply by drawing from the uniform distribution $\mathcal{U}(-0.01, 0.01)$ was found to be more effective. In both types of models, the biases were initialized to zero. Both FFNN and RNN models were trained using a learning rate of 10^{-5}, a gradient moving average decay factor of 0.9 and a machine epsilon of 10^{-6}. For both FFNN and RNN models, dropout is used after the hidden layer to further prevent overfitting, with $p_{dropout} = 0.5$. The gradients of the RNNs were clipped to lie between -100 and 100. For RNN models, the batch size is set to one piece (the maximal sequence length is 3808, the number of onsets in the longest piece in the dataset). For FFNNs, the batch size was set to 1000. All networks were trained for a maximum

Model	R^2	r
LBM	0.026	0.232
NBM (20)	0.060	0.249
RNBM (20)	**0.136**	**0.374**

Table 4.4: Predictive results for IOI, averaged over a 10-fold cross-validation on the Magaloff/Chopin corpus. Larger r and R^2 means better performance.

of 1000 epochs, stopping the training after 50 epochs without improvement in the loss function on the validation set.

4.3.6 Results and Discussion

The average of values of r and R^2 across all pieces in the Magaloff/Chopin dataset for each model type is presented in Table 4.4. Both measures suggest that the RNBM (20) gives a consistent improvement over both FFNN and linear models. Analysis of variance using linear mixed-effects revealed both a main effect of model on the squared error, and a significant interaction between model and fold on squared error. Because of this interaction, a post-hoc comparison (Tukey's HSD test) was performed per fold, rather than over all folds together. These tests show that the RNBM (20) models are significantly better than both NBM (20) and linear models on all folds ($p < 0.0001$). In seven folds, NBM (20) performed significantly better than LBM ($p < 0.0001$). In one fold, the linear model performed better than NBM (20) ($p < 0.0001$), and in two folds NBM (20) and LBM were not significantly different.

These results seem to confirm the respective benefits of non-linearity and temporal dependencies for modeling expressive timing in classical piano performances, as described in Sections 4.3.1 and 4.3.4. In the following, we focus on the latter aspect, highlighting learned interactions between the hidden states at different times, a capability that sets apart RNN-based RNBMs from FFNN-based NBMs and LBMs.

Sensitivity analysis

In order to analyze the effects of past and future score information on the performance of a score onset, we use a method called *differential sensitivity analysis* (Hamby, 1995). This method quantifies how strongly the output of the model changes in response to changes in each of the basis functions at each time step. The differential sensitivity of the output of the predictive function for an expressive parameter $f(\cdot\,;\boldsymbol{\theta})$ to the i-th basis function φ_i at the j-th score position o_j can be computed as

$$S_{i,j} = \mathbb{E}\left\{\frac{\partial}{\partial\varphi_i(o_j)} f(\boldsymbol{\varphi}(o_j)\,;\boldsymbol{\theta})\right\}. \tag{4.36}$$

Plotting the sensitivity values of a particular basis function at different time steps relative to the current time step yields a kind of *temporal profile* of the basis function, showing how that basis function affects an expressive parameter in its prior and posterior context. This is done in Figure 4.7 for log BPR, which shows the model sensitivity with respect to three basis functions, for the models trained in each of the ten folds of the cross-validation. The curves show how an activation of the basis function in the present (time step 0) affects the values of log BPR in the past (negative time steps), and in the future (positive time steps).

The different curves within each of the three plots correspond to the models trained on the 10 different folds of the cross-validation. Although there are differences, the overall shapes of the curves within a plot are very similar, showing that the temporal sensitivity profiles are not due to random factors (such as model initialization) that are irrelevant to the data.

A ranking of all 220 basis functions according to model sensitivity showed that the models are most sensitive to the fermata basis. Unsurprisingly, the models have learned from the data that the fermata sign indicates a lengthening of the note. However, the part of the plot before time step 0 shows that the fermata sign also affects the values of log BPR preceding it, provoking an incremental lengthening of the preceding values of log BPR up to the fermata sign itself. This result demonstrates the anticipatory function of expressive timing described by Istók et al. (2013).

A different pattern is exhibited for duration. Here there is a negative sensitivity of performed log BPR to the notated duration, implying that long notes are systematically shortened. Interestingly, the surrounding values of log BPR are lengthened slightly. This suggests that durational contrasts in the score are softened, a phenomenon also reported by Gabrielsson et al. (1983). Although both sharpening and softening of contrasts have been found to be a means of expressing emotions in music (Lindström, 1992), the models have not learned sharpening effects from the data. This may be due to the character of the music in the Magaloff/Chopin dataset.

Musical accents have a less marked, and more diffuse effect on log BPR. Models from different folds appear to have learned different regularities, although there is a weak tendency to slow down towards an accent, followed by a slight speed up following the accent.

4.4 Conclusions

Expressive performance of music is a complex phenomenon, of which our understanding is far from complete. In this chapter we have extended the simple LBMs discussed in Chapter 3 in two ways: Firstly, we introduced the NBM, which uses more powerful non-linear models (FFNNs), and second, we extended this non-linear model to be able to capture the sequential nature of music (using RNNs). For this purpose, we extended the way expressive parameters and score representations are computed.

We have carried out an extensive comparative evaluation of linear, non-linear and recurrent non-linear models on different corpora of classical piano performance. The results of these evaluations show that both NBMs discussed in Section 4.2 and RNBMs 4.3 allow for more accurate modeling of expressive parameters than the LBM models described in Chapter 3. Furthermore, RNBMs have been shown to lead to more accurate predictions for expressive tempo than NBMs (suggested by experiments in Section 4.3.5).

The results of the evaluation of the NBMs discussed in Section 4.2 show that non-linear methods allow for substantially more accurate modeling of expressive dynamics. A qualitative analysis of the trained NBMs reveals that FFNNs effectively learn interaction-effects between aspects of the musical score that linear models cannot capture. Through this analysis we also found that the models reproduce several regularities in expressive dynamics that have been individually found or hypothesized in other literature, such as that high notes should be played louder, or that musical grouping (as expressed by slur marks) is a determining factor for expressive dynamics. Thus, a contribution of the study documented in Section 4.2.3 is that it provides evidence for these findings, which are sometimes no more than a musical intuition or conjecture, based on two independent data sets.

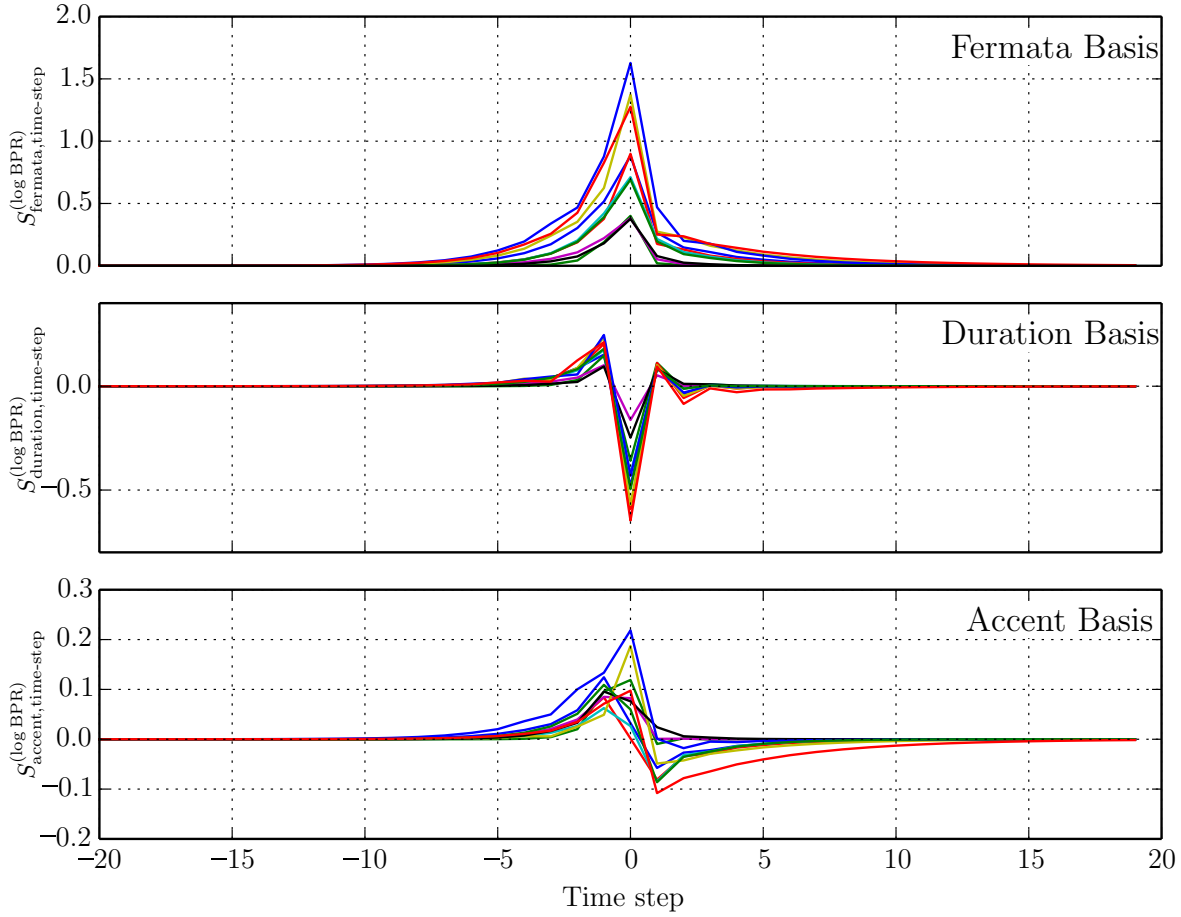


Figure 4.7: Sensitivity of log BPR predictions to Fermata, Duration and Accent basis functions, for the ten models obtained from 10-fold cross-validation. Positive values on the vertical axis denote a lengthening of BPR (tempo decrease), negative values a shortening (tempo increase). The curves show the effect of the activation of the basis function at time step $j = 0$ on log BPR values (see Equation (4.36)) in the past (negative time steps), and future (positive time steps).

The experiments reported in Section 4.3.5 show that the recurrent model predicts expressive timing much better than non-sequential versions of the model using the same score representations and performance representation. This is evidence that temporal dependencies do play an important role in expressive timing.

Further prospective work includes combining the current basis function modeling approach with (deep) unsupervised feature learning of musical score representations, following the work by Grachten and Krebs (2014) and van Herwaarden et al. (2014). The benefit of this hybrid approach is that it combines information about annotations in the musical score, that are not part of the representation learning process but are definitely relevant for modeling expression, with the adaptiveness and malleability of representations learned from musical data.

As previously stated in Section 3.2.4, a model with high predictive accuracy might not necessarily render a good musical performance of a piece. Therefore, in addition to numerical evaluation of the model outputs, future evaluation might also require listening tests. Nevertheless, we believe that the effort of systematically testing the perceptual validity of model outputs is best spent after the quantitative and data-oriented development of the model has stabilized.

An interesting intermediate form between numerical evaluation and perceptual validation of model outputs is to define perceptually inspired objective functions. In the computer vision and image processing communities there have been several efforts in this direction, including the definition of the structural similarity index (Wang et al., 2004), a perception-based metric that considers perceptual phenomena like luminance masking, as well as perceived change in structural information. For musical expression, a starting point is the perceptual model described in (Honing, 2006).

One of the limitations of the variants of the BMs presented in this chapter is that they do not account for different expressive interpretations of the same score. Future work might involve exploring probabilistic BMs that can explicitly learn multi-modal predictive distributions over the target values. Such models might be better suited to train on corpora consisting of multiple performances of the same piece (possibly by many different performers). In Appendix D we present preliminary results in this direction using Mixture Density Networks (Bishop, 2006; Graves, 2013), which use the output of neural networks to parametrize a probability distribution.

In most of this work we focus on analyzing and predicting expressive piano performances. Chapter 5 expands the model to the case of ensemble performance. Chapter 6 tests whether cognitively plausible features improve predictions of expressive dynamics and tempo. Chapter 7 presents an implementation of the BM framework discussed in this chapter for rendering expressive performances of musical pieces given their scores.

5 From Solo Piano to Orchestral Ensembles

The content of this chapter consists of work done in close collaboration with Thassilo Gadermaier and Maarten Grachten, in the frame of the Lrn2Cre8 and PHENICX projects. This chapter contains material published in:

- Gadermaier, T., Grachten, M., and Cancino-Chacón, C. E. (2016). Basis-Function Modeling of Loudness Variations in Ensemble Performance. In *Proceedings of the 2nd International Conference on New Music Concepts (ICNMC 2016)*, Treviso, Italy

  For this article, I designed and carried out the experimental evaluation.

- Grachten, M., Cancino-Chacón, C. E., Gadermaier, T., and Widmer, G. (2017). Towards computer-assisted understanding of dynamics in symphonic music. *IEEE Multimedia*, 24(1):36–46

  For this article, I designed and carried out the experimental evaluation, as well as devising and carrying out the differential sensitivity analysis method, which studied the contribution of score features to the output of the model and compared performances by different performers (the sensitivity difference graphs discussed in Section 5.4 below).

5.1 Introduction

The basis function modeling (BM) approach described in Chapters 3 and 4 has been developed for the purpose of modeling expression in solo piano performances, based on measurements obtained from a computer-controlled grand piano. In this chapter, we present an extension to the BM framework to make it suitable for modeling expression in orchestral music.

The contributions of this chapter are twofold. First, we extend the BM framework to accommodate ensembles of instruments, possibly including multiple instances of the same instrument, as is common in orchestral scores. We discuss the difficulties and complications that arise when dealing with recordings of large ensembles, rather than a single piano. Secondly, we propose the use of the model as a tool for explanatory purposes. The model can attribute variations in the expressive quality of a performance to factors like performance directives that were written in the score by the composer (like crescendo, diminuendo, and fermata), and other aspects of the written score, and could therefore be used to highlight differences between performances of the same piece. Explanatory visualizations of expressive performances based on this information could be used for didactic purposes, to introduce an audience to the phenomenon of expressive music interpretation. Such a tool may be part of an *active music listening interface* (Goto, 2007) for classical music such as the integrated prototype[1] of the PHENICX project (Liem et al., 2015).

The rest of this chapter is structured as follows. Section 5.2 discusses how to adapt the computation of the basis functions in an ensemble scenario. Section 5.3 presents an evaluation of

[1] http://beta.phenicx.com/

the proposed model for modeling expressive dynamics using the RCO/Symphonic dataset. Section 5.4 presents preliminary work using the extended BM framework as a tool for comparing expressive dynamics in symphonic music. Finally, conclusions are presented in Section 5.5.

5.2 Basis Functions for Orchestral Music

There are several issues to be dealt with in order apply the BM approach to orchestral performances. In the rest of this section, we will discuss these issues, and provide solutions.

5.2.1 Measured vs. Computed Expressive Parameters

In a piano, the degrees of freedom for sound production (and therefore expressive performance) are limited to only a few, well-defined dimensions: the piano key, the hammer velocity of the key mechanism, the timing of key press and key release and the pedals[2]. These dimensions can be measured relatively easily. As a result, it is possible to obtain precise measurements of piano performances in the dimensions mentioned above, through the use of computer-controlled pianos (e.g. see (Moog and Rhea, 1990) for a description of the mechanism and capabilities of the Bösendofer SE, a high quality computer controlled piano, and (Goebl and Bresin, 2003) for an evaluation of recording and reproducing precision of the Bösendorfer SE and the Yamaha Disklavier).

Similar measurements are typically not readily accessible for other classes of instruments, such as bowed string instruments or wind instruments, which have more complex sound production mechanisms. For example, although rich descriptions of performances of other instruments may be obtained with appropriate sensors (for instance designed to measure the bending of the reed in wind instruments (Hofmann et al., 2013) or bow movements in violin playing (Schoonderwaldt and Demoucron, 2009)), the usage of such sensors is often intrusive, and thus limited to experimental setups. Moreover, data recorded in this way is prone to noise, and bulky in the case of large ensembles.

An alternative to measuring aspects of the sound production of an instrument is to compute expressive information from audio recordings of professional music performances. Loudness and tempo can be obtained this way. Such audio-based measures yield a single value for each expressive parameter at each time instant, which is in contrast to measures obtained from computer-controlled pianos, which allow expressive parameters to be defined for individual notes, even if they occur at the same time instant. With audio-based measures, it is not possible to define onsets or loudness values for individual voices or instruments with sufficient precision – again, in contrast to MIDI-based measures[3].

Performance-score matching, which, for piano music, has been discussed in previous chapters, remains a challenging and laborious task for orchestral music. This is due in part to the complexities of onset detection for non-percussive instruments and the difficulties involved in obtaining high-quality digital versions of orchestral scores. The RCO/Symphonic dataset used in this chapter was prepared as part of the PHENICX project.

[2]For performances in MIDI format these dimensions correspond to the MIDI note number, the MIDI velocity, the onset and offset times of the note and the pedal information.

[3]Separation of audio into individual tracks by instrument is an active field of research in MIR, but at present remains an unsolved task.

5.2.2 Indexing Basis Functions

The BM approach is designed to generate a set of basis functions (see basis functions defined in Sections 3.3.2, 3.4.1 and 4.2.1), given a score $\mathfrak{S}$ for a single instrument. When training an expression model on a dataset containing performances of multiple pieces, the basis functions produced for each piece must be mapped to each other. For example, if we have two pieces, A and B, and the scores for both pieces contain dynamics markings $\boldsymbol{p}$ and $\boldsymbol{f}$, we assume that there is a single basis function $\varphi_{\boldsymbol{p}}$ and a single basis function $\varphi_{\boldsymbol{f}}$, instead of defining basis functions $\varphi_{\boldsymbol{p};\mathrm{A}}$ and $\varphi_{\boldsymbol{f};\mathrm{A}}$ that encode the markings $\boldsymbol{p}$ and $\boldsymbol{f}$ for piece A and *another* group of basis functions $\varphi_{\boldsymbol{p};\mathrm{B}}$ and $\varphi_{\boldsymbol{f};\mathrm{B}}$ for piece B. In this case, we say that both $\varphi_{\boldsymbol{p};\mathrm{A}}$ and $\varphi_{\boldsymbol{p};\mathrm{B}}$ are mapped to $\varphi_{\boldsymbol{p}}$; while $\varphi_{\boldsymbol{f};\mathrm{A}}$ and $\varphi_{\boldsymbol{f};\mathrm{B}}$ are mapped to $\varphi_{\boldsymbol{f}}$.

In the solo instrument setting, this mapping is done on the basis of labels that are uniquely assigned to each basis function (in the example above, the labels are $\boldsymbol{p}$ for basis function $\varphi_{\boldsymbol{p}}$ and $\boldsymbol{f}$ for basis function basis function $\varphi_{\boldsymbol{f}}$, since these basis functions encode dynamics markings $\boldsymbol{p}$ and $\boldsymbol{f}$, respectively). In orchestral scores, however, the labels are not unique any longer, since the same basis functions are produced for each instrument (coding the same type of score information, but for different instruments). For example, in an orchestral score we could have different dynamics markings for each instrument, for example there could be an $\boldsymbol{mf}$ in the flute while at the same time an $\boldsymbol{f}$ in the cello section. To deal with this, it is necessary to index the basis functions by the tuple *(instrument name, basis function label)*, i.e. $\varphi_{\textit{instrument; label}}$. Continuing with the previous example, this indexing would result in basis functions $\varphi_{\mathrm{Flute};\,\boldsymbol{mf}}$ and $\varphi_{\mathrm{Violoncello};\,\boldsymbol{f}}$.

A further issue is that the notated instrument names in the score do not follow any strict standard. Instrument names may be written in different languages (e.g. “Fagott”, “bassoon”), and may be abbreviated (e.g. “Vln.” for violin, “Cl.” for clarinet). To overcome this issue, the instrument names and abbreviations extracted from the score are matched to one of a set of *canonical instrument names* (the unabbreviated English names), using string matching techniques. This name matching is illustrated in Figure 5.1, where the names of the instruments in the score in the left are in Italian, and the names of the instruments of the score in the right are in German, which are matched to the names of the instruments in English.

5.2.3 Merging and Fusion of Basis Functions within Instrument Classes

In orchestral scores, there may be several instances (voices) of an instrument, usually designated by numbers (e.g. “Violin 1”, “Violin 2”). Furthermore, multiple instances of an instrument may share a single staff. This is shown in Figure 5.1.

The occurrence of multiple instruments of the same type poses a problem for training the model, since it is not clear how the mapping of basis functions across pieces should be defined in order to create a consistent dataset consisting of multiple pieces. For instance, when one piece involves a single horn, and another piece involves two horns, the question which of the two horns in the latter piece should be mapped to the horn in the first piece is arbitrary, and moreover, it is unclear how to deal with the remaining, unmapped horn.

For this reason, we choose to combine all instances of the same instrument class into a single set of basis functions, using a *fusion operation* that can be specified per basis function type. In this way, for each piece there is a single set of basis functions conveying the activity of a given instrument *class*, rather than one set for each *instance* of that class.

In Figure 5.2, we give an example: two staves are shown where the first staff holds a score part

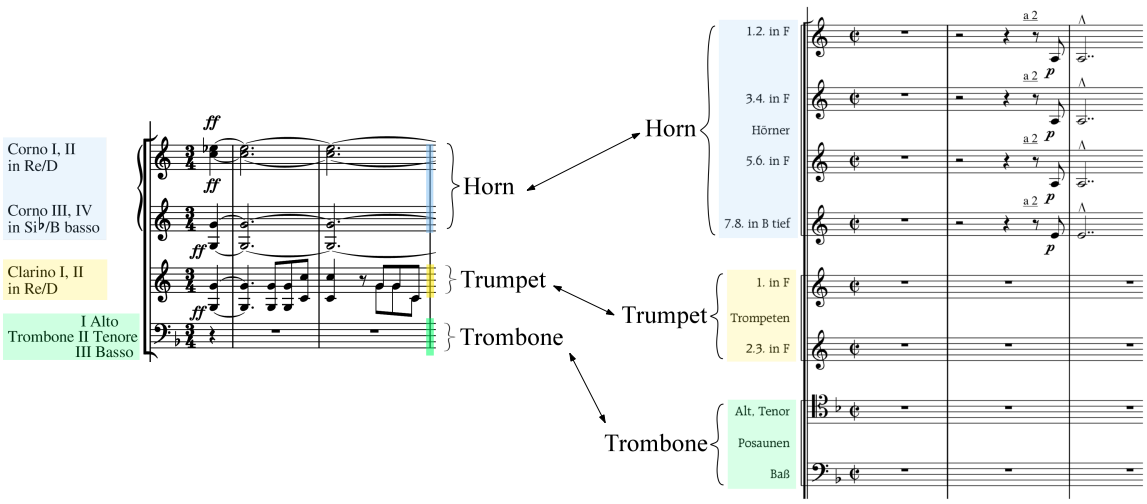
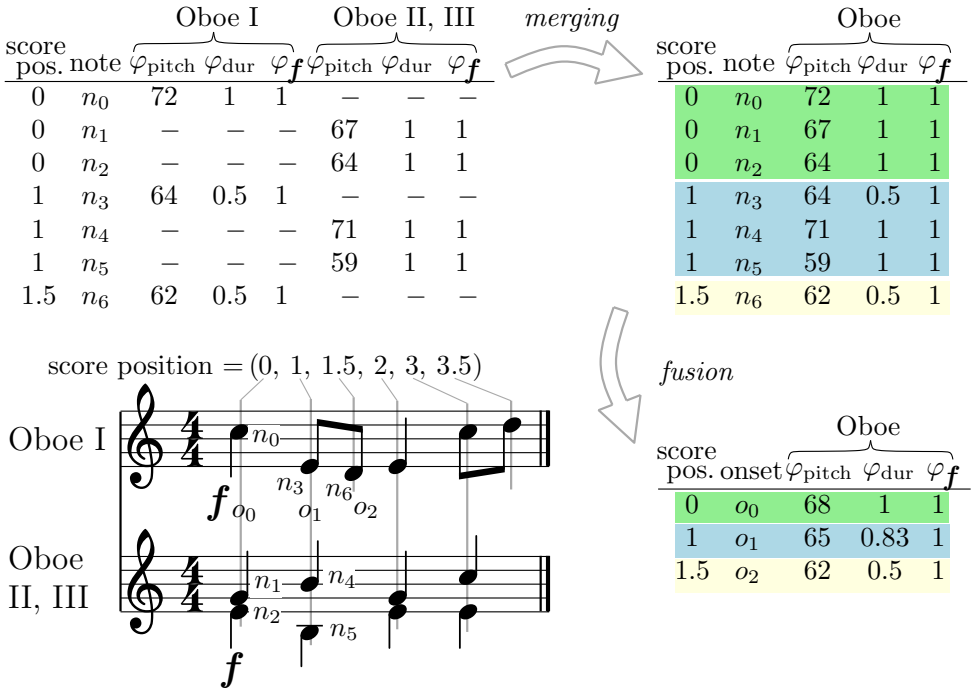


Figure 5.1: Motivation for *merging* and *fusion*. Different pieces' scores will in general have different number of instances of the same instrument class (e.g. 4 vs. 8 Horns). Carefully note the different possibilities to distribute voices across staves: in the right hand score excerpt, the lower staff of the trumpet carries two instances, whereas the upper staff only one. Similar for the trombone. The instances of one instrument class (e.g. 8 horns, right) will be *merged* to one representative to be matched to the representative from the other score (subsuming 4 horns, left).

consisting of a single instance (Oboe I). The second staff is a score part with two instances of the same instrument (Oboe II, III). For each score note n_i a set of basis functions is extracted from the score.

Following the extraction, the per score part defined basis functions will be combined to per instrument class basis functions. This works as follows: The values of the basis functions associated with each note form a row of a matrix (top left matrix in Fig. 5.2). The basis functions associated with each score part occupy their own set of columns in this matrix. The rows are sorted according to the respective associated notes' onset times. *Merging* compresses this matrix to another matrix (shown top right) where the columns already represent per instrument class basis functions, while still having (possibly) multiple rows per time stamp. *Fusion* now combines each subset of rows with the same time stamp into a single row by subsuming each column's values (within the row subset) into a single value, as is shown by the bottom right matrix. This is achieved with a fusion operation fusion$(\cdot)$ that can be specified separately for each basis function type (i.e. per column). Examples for fusion operations are the max$(\cdot)$ function that takes the maximum value across the subset of rows, the min$(\cdot)$ (taking the minimum value) and the avg$(\cdot)$ (taking the average) functions. In the example in Fig. 5.2, the average of the values was taken for each column.

A disadvantage of the merging and fusion of basis functions within instrument classes described in this section is that this procedure does not directly take into account genre conventions regarding certain instruments, for example, the usual complementary roles that the Violin I and II sections play in symphonic music of the Classical and Romantic Periods.



score pos.	note	Oboe I φ_{pitch}	Oboe I φ_{dur}	Oboe I $\varphi_{\boldsymbol{f}}$	Oboe II, III φ_{pitch}	Oboe II, III φ_{dur}	Oboe II, III $\varphi_{\boldsymbol{f}}$
0	n_0	72	1	1	–	–	–
0	n_1	–	–	–	67	1	1
0	n_2	–	–	–	64	1	1
1	n_3	64	0.5	1	–	–	–
1	n_4	–	–	–	71	1	1
1	n_5	–	–	–	59	1	1
1.5	n_6	62	0.5	1	–	–	–

score pos.	note	Oboe φ_{pitch}	Oboe φ_{dur}	Oboe $\varphi_{\boldsymbol{f}}$
0	n_0	72	1	1
0	n_1	67	1	1
0	n_2	64	1	1
1	n_3	64	0.5	1
1	n_4	71	1	1
1	n_5	59	1	1
1.5	n_6	62	0.5	1

score pos.	onset	Oboe φ_{pitch}	Oboe φ_{dur}	Oboe $\varphi_{\boldsymbol{f}}$
0	o_0	68	1	1
1	o_1	65	0.83	1
1.5	o_2	62	0.5	1

Figure 5.2: Illustration of *merging* and *fusion* of score information of two different parts belonging to the same instrument class "Oboe". The first matrix (top left) shows three example basis functions, φ_{pitch}, $\varphi_{\text{duration}}$, and $\varphi_{\boldsymbol{f}}$, for the first few notes of each of the two score parts. Note the interleaved, consecutive layout of simultaneously occurring notes, as indicated by the first column of each matrix, giving the notes' onset times. The matrix top right illustrates *merging*, where the data of both score parts were combined to give one column per basis function. Finally, the third matrix is the result of *fusion* operations, applied per basis function to each set of values occurring at the same time point. The fusion operation used takes the average of the respective values. See text for further explanation.

5.2.4 Aggregation of Basis Functions of Instrument Classes in a Piece

After collecting per instrument class basis function matrices, we need one final step to conclude the data extraction for a piece p. All instrument classes' data are aggregated into a single per-piece matrix $\mathbf{\Phi}_p$. This is schematically shown in Figure 5.3.

5.3 Experiments: Expressive Dynamics

To assess the predictive accuracy of the model, we conduct a leave-one-out cross validation experiment to see how well the BMs predict expressive dynamics. We use the RCO/Symphonic dataset described in Section 2.4.4. As previously mentioned, this dataset consists of recordings of symphonies from the Classical and Romantic period performed by the Royal Concertgebouw Orchestra. The process of extracting basis functions generates 1420 basis functions post merging and fusion, which are listed in Appendix A.2.

score pos.	Oboe φ_{pitch}	Oboe φ_{dur}	Oboe $\varphi_{\boldsymbol{f}}$	Clarinet φ_{pitch}	Clarinet φ_{dur}	Clarinet $\varphi_{\boldsymbol{f}}$	Bassoon φ_{pitch}	Bassoon φ_{dur}	Bassoon $\varphi_{\boldsymbol{f}}$	
0	68	1	1	64	1	0	0	0	0	$\cdots$
1	65	0.83	1	62	0.83	0	0	0	0	$\cdots$
1.5	62	0.5	1	59	0.5	0	0	0	0	$\cdots$
$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	$\vdots$	

Figure 5.3: Aggregating the per instrument class matrices into a per piece matrix. Note how the Clarinet has different dynamics prescribed (so that the ***forte*** basis function is all zeros). The Bassoon is obviously silent in this part of the piece.

5.3.1 Expressive Parameter

For computing expressive dynamics, we use a perceptual measure of the instantaneous loudness, i.e.

$$\mathsf{y}_{\text{loud}}(o_i) = \frac{\text{loudness}(o_i) - \text{loudness}_{mean}}{\text{loudness}_{std}}, \tag{5.1}$$

where $\text{loudness}(o_i)$ returns the EBU R 128 loudness measure (EBU-R-128, 2011), associated with the i-th onset in score $\mathfrak{S}$, and loudness_{mean} and loudness_{std} are the mean and the standard deviation of the values of the loudness per piece, respectively. The EBU R 128 is a loudness measure defined by the European Broadcasting Union that takes into account human perception, (i.e., the fact that signals of equal power but different frequency content are not perceived as being equally loud) and is the recommended way of comparing loudness levels of audio content in the broadcasting industry. Because we want to focus on variations in loudness, rather than the overall loudness level and range, we subtract the mean and divide by the standard deviation of the loudness values per piece.

5.3.2 Model Architectures and Training

In these experiments we compare the linear, non-linear and sequential versions of the BM framework.

The NBM used in this experiment consists of an FFNN with a single hidden layer with 20 rectified linear units, and a linear output layer with a single unit. The RNN of the RNBM consists of a single bidirectional vanilla recurrent layer with 20 rectified linear units and a linear output layer with a single unit. The bidirectional recurrent layer of this architecture uses a linear combination of the outputs of the forward and backward layers, as described in Equation (4.31), with $\alpha_f = \alpha_b = 1$. Both FFNN and RNN architectures used in this evaluation are identical to the architectures used for the experiments described in Section 4.3.5.

The LBM models were trained using LSMR (Fong and Saunders, 2011). Both versions of the non-linear models (NBM and RNBM) were trained using RMSProp (Tieleman and Hinton, 2012) with a learning rate of 10^{-5}, a gradient moving average decay factor of 0.9 and an epsilon for numerical stability of 10^{-6}. The weights of NBM models were initialized using the method proposed in Glorot and Bengio (2010), while the weights of RNBM models were randomly sampled from $\mathcal{U}(-0.01, 0.01)$. All biases and initial states were initialized to zero. To avoid overfitting we used dropout on the hidden layers of neural network models with a probability of 0.5. For RNN models we truncate the back-propagated gradients after 200 time steps. The gradients were clipped to lie between -100 and 100 to avoid numerical instabilities. From the

19 training pieces, four pieces were kept for validation, and the training was stopped if there was no further improvement in the loss on this validation set after 100 epochs. All models were trained for a maximum of 2000 epochs.

5.3.3 Predictive Accuracy

For each piece, we report the coefficient of determination (R^2), measuring the proportion of variance in the recorded loudness curve that is explained by the model, and Pearson's correlation coefficient (r), measuring the strength of the linear dependence between the recorded and the predicted loudness curves. The results of the experiments are shown in Table 5.1. We observe that both the R^2 and the r values for LBM are generally lower than those for NBM and RNBM, demonstrating that the non-linear modeling provides a clear advantage over the linear modeling approach. We conducted a one-way ANOVA level to test the difference in R^2 for LBM, NBM and RNBM. There was a clear outlier for the R^2 for the first movement of Bruckner's Symphony No. 9 for the LBM, which we excluded from the analysis. It should be noted that this was the only model that produced such an outlier, which is in line with our evaluation of LBMs as less accurate than NBMs and RNBMs. There was a significant difference at the $p < 0.01$ level as measured by Fisher's F ratio ($\mathrm{F}(2, 54) = 5.47, p = 0.007, \eta^2 = 0.17$). Furthermore, we conducted a paired samples one-tailed t-test to compare the difference in R^2 for NBM and RNBM models. There was a significant difference at the $p < 0.01$ level ($\mathrm{mean} = 0.05, \mathrm{std} = 0.04$); $\mathrm{t}(18) = 5.69, p < 0.001$, Cohen's $d = 0.75$). Given the relatively small data set, this result is not trivial, since the NBM and RNBM have many more parameters than the LBM model, and are therefore more prone to overfitting. Furthermore, the RNBM model provides more accurate predictions than the NBM model, although this advantage is less prominent than the advantage over LBM.

The fact that the results of the linear model are inferior suggests that although the basis functions used to represent the score capture relevant information, their shapes (such as the ramp function to represent a ***crescendo***) are too schematic to work well as approximations of measured loudness curves. The improvement of the results in the NBM and RNBM models suggest that the non-linear transformation of these shapes alleviates this problem to some extent. The capability of the non-linear models of modeling interactions between basis functions, as demonstrated in (Cancino Chacón and Grachten, 2015) (described in Section 4.2.3), may further explain the improved results of these models.

5.4 The BM as an Analytical Tool for Dynamics in Symphonic Music

In this section, we demonstrate that the BM framework can be used for explanatory purposes, and thus form the basis for a tool that elucidates differences in expressive interpretations between performances.

The explanatory power of BM models lies in the fact that they represent dynamics as a function of the basis functions. As a model learns from training data how the basis functions relate to dynamics, some basis functions may prove to be very important for an accurate prediction of dynamics, while others may have little or no influence at all. In other words, the model learns to be more *sensitive* to some basis functions than to others. We can impose sensitivities specific to a particular performance on a model by *fitting* the model to that performance—adjusting its parameters such that its prediction error for the dynamics of that performance is minimized. When fitting models to two different performances of a piece, the differences in dynamics between the performances tend to lead to different sensitivities in the models. For

Composer / Piece		R^2			r		
		LBM	NBM	RNBM	LBM	NBM	RNBM
Beethoven S5	Mv 1	-0.26	-0.22	**-0.18**	0.18	0.21	**0.26**
	Mv 2	0.34	0.46	**0.56**	0.58	0.70	**0.76**
	Mv 3	0.23	0.40	**0.44**	0.53	0.64	**0.66**
	Mv 4	0.06	**0.26**	0.25	0.41	**0.53**	0.52
Beethoven S6	Mv 1	0.36	0.36	**0.39**	0.61	0.63	**0.65**
	Mv 2	0.07	0.15	**0.17**	0.36	0.40	**0.41**
	Mv 3	0.51	0.60	**0.62**	0.72	0.81	**0.82**
	Mv 4	0.11	0.27	**0.29**	0.38	0.54	**0.56**
	Mv 5	0.36	0.44	**0.49**	0.60	0.70	**0.75**
Beethoven S9	Mv 1	0.34	0.36	**0.42**	0.59	0.61	**0.65**
	Mv 2	0.36	0.40	**0.53**	0.60	0.64	**0.74**
	Mv 3	-0.30	-0.06	**-0.02**	0.20	0.17	**0.22**
	Mv 4	0.11	0.37	**0.49**	0.52	0.64	**0.70**
Mahler S4	Mv 1	-0.17	0.29	**0.37**	0.37	0.54	**0.61**
	Mv 2	-0.48	-0.02	**-0.02**	0.06	0.20	**0.23**
	Mv 3	-1.22	0.25	**0.26**	0.20	0.51	**0.53**
	Mv 4	-1.99	0.09	**0.18**	0.15	0.33	**0.44**
Bruckner S9	Mv 1	-39.06	0.45	**0.59**	0.26	0.68	**0.77**
	Mv 2	0.24	0.48	**0.55**	0.58	0.72	**0.74**
	Mv 3	-3.54	0.32	**0.40**	0.25	0.57	**0.65**

Table 5.1: Predictive accuracy in a leave-one-out scenario for the different models. For both R^2 and r, larger is better. Best value per piece and measure emphasized in bold.

example, a model fitted to a dramatic performance may learn that dynamics annotations such as ***piano*** and ***forte*** have a large effect on the dynamics of the performance, whereas a model fitted to a more restrained performance may be less sensitive to these annotations. Thus, comparing differences in sensitivities between models fitted to different performances can give us qualitative explanations of the differences in dynamics, in the style of "Performance A is louder than performance B at this point in the piece, because the string instruments are more prominent", or "Performance B emphasizes the downbeat more strongly than Performance A".

When fitting two models to two different performances for comparison purposes, it makes sense to start the fitting process from a common model that was *pretrained* on a number of other recordings. Firstly, this speeds up the fitting process since the pretrained model will already provide a rough approximation of the dynamics curves, and secondly, starting from a common basis encourages similar explanations for similar trends in the performances, and thereby parsimonious explanations of the differences between the performances.

We compute the sensitivities of a model to each of the basis functions using a local differential–based sensitivity analysis technique (Hamby, 1995), which consists of computing the gradient of the output of the model with respect to each of its inputs. By multiplying the gradients (sensitivities) with the inputs (basis functions) over the course of a piece, we obtain a *sensitivity graph* for a performance. The multiplication is motivated by the fact that even when a model is sensitive to a particular basis function, this basis function does not affect the output of the model whenever it is inactive (i.e. zero valued). The sensitivity graph for a performance can be thought of as a matrix $\boldsymbol{SG} \in \mathbb{R}^{N_o \times M}$, with N_o the number of onsets in the score $\mathfrak{S}$ and M the number of basis functions used to represent such a score. The (i, j)-th element in this matrix,

$SG_{i,j}$, is given by

$$SG_{i,j} = \frac{\partial}{\partial \varphi_i(o)} f^{(\text{loud})}(\boldsymbol{\varphi}(o))\Big|_{o=o_j} \times \varphi_i(o_j), \tag{5.2}$$

where $f^{(\text{loud})}(\cdot)$ is the output of the neural network predicting loudness, and $\varphi_i(o_j)$ and $\boldsymbol{\varphi}(o_j)$ are the values of the i-th basis function and the values of all basis functions for score onset o_j, respectively. Moreover, by subtracting the sensitivity graphs of two performances, we obtain what we call a *sensitivity-difference (SD) graph*, a visual representation expressing the relative influence of each basis function in each of the two performances. Such a graph is given by

$$SD = \boldsymbol{SG}^{\text{T}}_{\text{perf 1}} - \boldsymbol{SG}^{\text{T}}_{\text{perf 2}}, \tag{5.3}$$

where $\boldsymbol{SG}^{\text{T}}_{\text{perf 1}}$ and $\boldsymbol{SG}^{\text{T}}_{\text{perf 2}}$ are the transposed sensitivity graphs of each performance to be compared[4].

As a case study, we compare performances of the 3rd Movement (*Lustiges Zusammensein der Landleute*) of Beethoven's Symphony No. 6 Op. 68 by different conductors and orchestras. This piece is a scherzo which suggests dances of the country folk. The scherzo is in a $\frac{3}{4}$ meter, with its trio in a $\frac{2}{4}$ meter. The two performances we compare here are by the conductors Georg Solti (with the Chicago Symphony Orchestra, recorded in 1974) and Nikolaus Harnoncourt (with the Chamber Orchestra of Europe, recorded in 1991), hereafter referred to as Solti and Harnoncourt, respectively. We compare the performances by using a RNBM model that was pretrained on the dataset described in Table 2.3, and then fitting the pretrained model to both performances, respectively.

As an example, consider the small, but marked difference between Solti and Harnoncourt in bars 87-90 of the SD graph in Figure 5.4 (showing a selection of the most influential basis functions in the fragment). In bar 87, Beethoven's score prescribes a four-bar ***diminuendo*** of the violins to transition from an ongoing ***fortissimo*** (***ff***) passage (starting before and continuing in the depicted fragment) to a quiet and lyrical ***pianissimo*** (***pp***) passage featuring a singing oboe, starting with bar 91. The SD graph shows that the increased loudness in Harnoncourt (compared to Solti) is attributed by the model to a sustained influence of the ***ff*** in the violins over the course of the diminuendo. Note that this attribution is a parsimonious explanation of the loudness difference, because it involves only a single basis function. A hypothetical, less concise explanation, for instance, could involve an increased influence of each of the metrical positions for Harnoncourt. Together, the increased influence of these basis functions would also lead to a louder performance of the fragment overall, but may not be compatible with Harnoncourt's interpretation of the rest of the piece.

Listening to the respective passages, we note indeed a clearly audible difference: Solti takes the ***diminuendo*** very strictly, immediately softening the orchestra and quickly arriving at a very soft playing level before the actual arrival of the ***pp***. Harnoncourt's ***ritardando*** is more of a continuation of the preceding ***fortissimo*** passage: he only grows slightly softer during the ***ritardando***, and obeys the ***pp*** more abruptly when it arrives (the purple color of the ***pp*** starting with bar 91 indicates that Solti's ***pp*** is actually slightly louder than Harnoncourt's). It turns out that these are consistent and obviously deliberate choices, as we find the exact same pattern later on in the piece, in bars 292–295, where we have an analogous musical passage.

Furthermore, the SD graph shows a slight but systematic pattern in the metrical basis functions. This pattern suggests the model found slight differences in the metrical accentuation, with Solti

[4]The reason for the transposition is that, when plotted, it is more intuitive to visualize the temporal dimension in the x axis and the basis functions in the y axis.

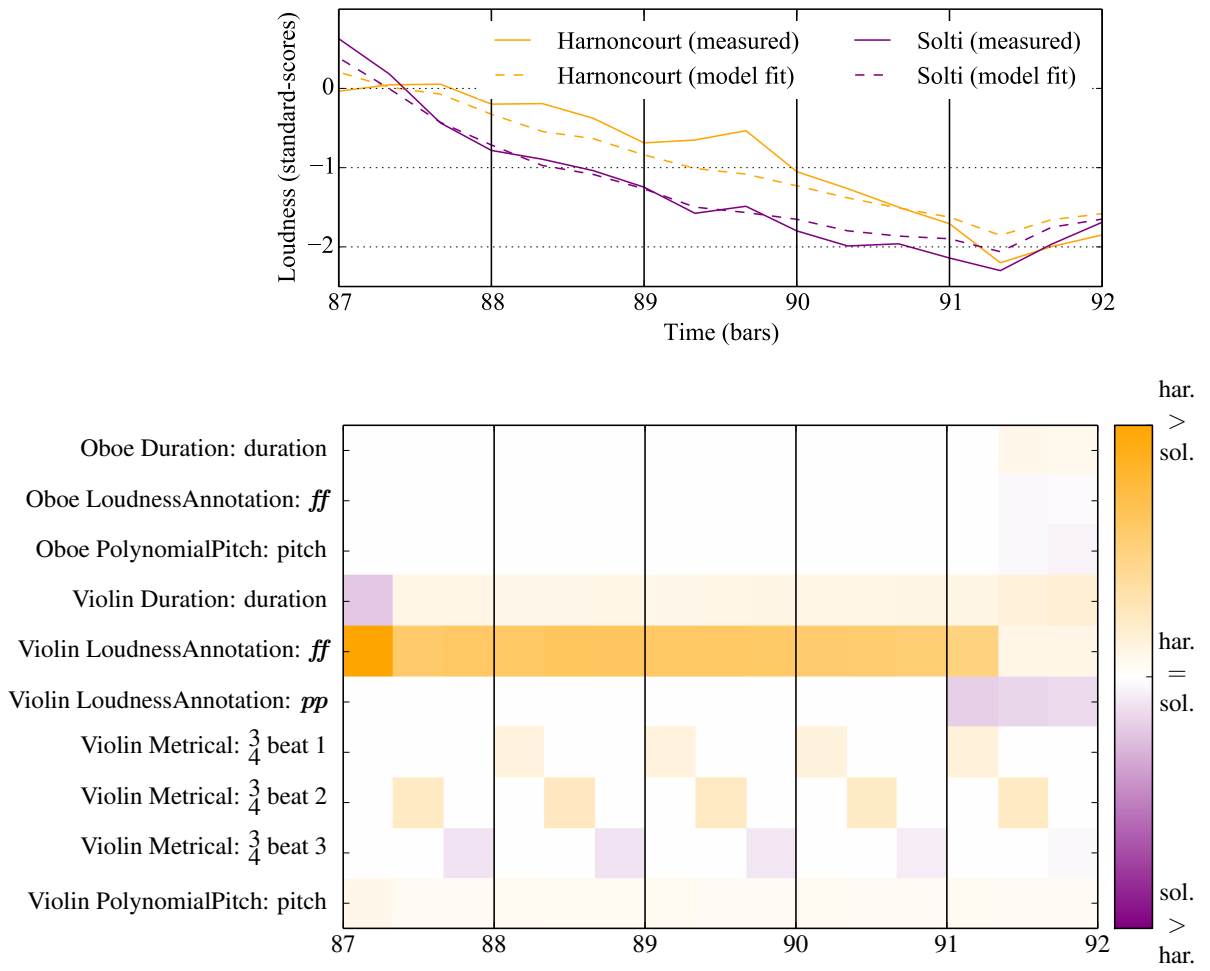


Figure 5.4: Top: Measured and fitted loudness curves for an excerpt of the performance of Beethoven's 6th Symphony, 3rd Movement (bars 87 to 92) by Harnoncourt (orange) and Solti (purple). Bottom: sensitivity-difference graph for Harnoncourt and Solti. Orange tones indicate that a basis function has a stronger (positive) contribution to loudness in Harnoncourt than in Solti; purple tones indicate the opposite.

placing more emphasis than Harnoncourt on the last beat of the bar, and vice versa for the first to beats. Listening reveals that these differences are too subtle to be heard, however.

Finally, it is important to note that the SD graph pertains to the *model fit* dynamics curves in Figure 5.4 (top), not the *measured* curves. There are some fluctuations in the measured curve (such as that on beat 3 of measure 89) that are not captured, and therefore cannot be explained by the SD graph.

5.5 Conclusions

This chapter extends the BM framework for modeling loudness variations in audio recordings of symphonic pieces. An evaluation of different variants of the model shows that a non-linear version including temporal dependencies is most effective in a predictive setting, where the model predicts loudness variations based on the written score, after being trained on a set of recordings. Examples given in Section 5.4 illustrate how the model can be used as a way to

explain differences between performances in terms of the written score.

It must be kept in mind however, that the data set used for validating the model, although comprising works from several composers, is performed by a single orchestra, and two different conductors. Preliminary cross-validation suggests the model trained on RCO recordings generalizes well to recordings by different orchestras, but more elaborate experimentation is necessary to make stronger claims about the robustness of the model against variance in recording/mixing/mastering conditions across recordings.

Furthermore, the measured overall loudness variation is only a coarse measure of (a single aspect of) musical expression, and currently, the model approximations (and thus its explanations) may not be adequate at all positions in the performance. Better model approximations and predictions will allow for novel explanatory uses, such as using the predictions of a model that was trained on multiple performances of a piece as a "baseline" performance, based on which the idiosyncrasies of conductors can be established.

The examples were hand-picked here, since the expression model is currently in a stage where we are testing its validity, and experimenting with different sets of basis functions. In the future, the model should be capable of automatically identifying excerpts from a piece where two or more performance differ substantially from each other, in order to highlight them to the listener, and show which aspects of the performance are different.

In combination with a web-service for aligned music playback and visualization, such as presented by Gasser et al. (2015), the model presented here allows listeners with a desire to get a better grasp on a piece of music, to compare different performances of the piece in terms of their expressive character, and get a better understanding of what it is that makes the performances different.

6 Understanding the Role of Cognitively Motivated Features in Predictions of Expressive Performances

This chapter contains material published in

- Cancino-Chacón, C., Grachten, M., Sears, D. R. W., and Widmer, G. (2017c). What Were You Expecting? Using Expectancy Features to Predict Expressive Performances of Classical Piano Music. In *Proceedings of the 10th International Workshop on Machine Learning and Music (MML 2017)*, Barcelona, Spain
- Cancino-Chacón, C. and Grachten, M. (2018). A Computational Study of the Role of Tonal Tension in Expressive Piano Performance. In *Proceedings of the 15th International Conference on Music Perception and Cognition (ICMPC15 ESCOM10)*, Graz, Austria

6.1 Introduction

Expressive performance of music constitutes an important part of our enjoyment of several kinds of music, including Western art music and jazz. In these kinds of music, an expressive performance is not expected to be an exact mechanical rendition of what is written in the score, but a performer's interpretation of both the intentions of the composer and her or his own intentions that are conveyed to the listener through variations in dimensions such as tempo, dynamics and timbre. As discussed in Chapter 2, computational models of musical expression can be used to explain the way certain properties of a musical score relate to an expressive rendering of the music (Widmer and Goebl, 2004). However, existing models, including the BM approach discussed in Chapters 3 and 4, tend to use a combination of high- and low-level *hand-crafted features* reflecting structural aspects of the score that might not necessarily be perceptually/cognitively relevant features.

Previous work has shown a relation between such expressive variations and perceptual characteristics derived from the musical score, such as musical expectations, and perceived tension (Meyer, 1961; Huron, 2006; Farbood, 2012; Chew, 2016; Gingras et al., 2016). According to the increasingly prominent predictive coding paradigm (Friston and Kiebel, 2009; Clark, 2013), the brain is essentially a *prediction machine*, aiming to minimize the discrepancy between an organism's expectancies and imminent events. This view also has implications for music listening: it can be argued that musical structures set by the composer generate expectations in the listener of how the music will continue, and it is the role of the performer to help the listener understand the composer's intentions. For example, a performer might decrease the listener's uncertainty about future events by slowing down at unexpected/uncertain moments, or speeding up at expected/certain ones, thereby decreasing the processing burden on listeners during perception.

In this chapter we study the role of two types of perceptually relevant features, musical expectations and tonal tension, in the prediction of expressive music performance, in particular

expressive tempo and dynamics. The rest of this chapter is organized as follows: Section 6.2 presents a brief overview on related work quantifying musical expectation and tonal tension. Section 6.3 presents a formalization of expressive parameters for expressive tempo and dynamics, describes the expectancy, tension and score features employed in this study, and finally outlines the predictive model used to estimate the expressive parameters. Section 6.4 describes the empirical evaluation of the proposed approach, the results of which are discussed in Section 6.5. Finally, conclusions are stated in Section 6.6.

6.2 Related Work

In this section we present a brief description of related work on quantifying musical expectation and tonal tension in the context of computational models of expressive performance. A more thorough discussion is presented in Section 9.3.3.

6.2.1 Musical Expectation

To examine the relationship between the formation of expectations during music listening on the one hand, and the realization of musical performances on the other, Gingras et al. (2016) employed the *Information Dynamics of Music* model (or IDyOM) (Pearce, 2005), a probabilistic model of auditory expectation that computes information-theoretic features relating to the prediction of future events. In their study, these information-theoretic features were shown to correspond closely with temporal characteristics of the expressive performance, which suggests that the performer attempts to decrease the processing burden on listeners during perception by slowing down at unexpected/uncertain moments and speeding up at expected/certain ones.

Here we present preliminary work to support the claim that expectancy measures can inform predictions of expressive parameters related to tempo and dynamics. We extend the work in (Gingras et al., 2016) in two ways. First, rather than simply demonstrating that expectancy measures are related to expressive performances, we show that the use of expectancy features improves the predictive quality of models using other score descriptors, thus providing a more comprehensive framework for the modeling of expressive performances in music of the common-practice period. Second, as opposed to *fitting* the expectancy features to each performance (i.e. training and testing the model on the same performance), the models presented in this paper are evaluated by measuring their prediction error on unseen pieces.

6.2.2 Tonal Tension

The concept of musical tension is highly complex and multidimensional, and thus, difficult to formalize or quantify (Farbood, 2012; Herremans and Chew, 2016). Informally, *"increasing tension can be described as a feeling of rising intensity or impending climax, while decreasing tension can be described as a feeling of relaxation or resolution"* (Farbood, 2012, pp. 387). The music cognition literature has shown that aspects related to musical tension include both psychological factors such as expectation and emotion; and musical factors such as rhythm, timing and dynamics and tonality. For a more thorough description of aspects that contribute to musical tension, we refer the reader to (Farbood, 2012) and references therein.

In this work we use a computational approach to study the role of tonal tension features – as proposed by Herremans and Chew (2016) – in the prediction of expressive performances of

Classical piano music. Computational models of musical expression can be used to explain the way certain properties of a musical score relate to an expressive rendering of the music (Widmer and Goebl, 2004). The KTH model (Friberg et al., 2006), one of the most important *rule-based* models of expressive performance includes three rules that take into account tonal tension:

1. *melodic charge*, which emphasizes notes that are tonally distant from the root of the current chord (i.e. far away in the circle of fifths);
2. *harmonic tension*, which emphasizes chords that are tonally distant relatively to the key of the piece; and
3. *chromatic charge*, which emphasizes regions in which the melody consists of with relatively small intervals.

This emphasis is performed through gradual increase in loudness and decrease in tempo towards the emphasized regions (Friberg et al., 2006). Melodic and harmonic charges are relevant for traditional harmony, since they relate to the relative consonance/dissonance of a chordal structure given a tonal context. Chromatic charge, on the other hand, is more relevant for atonal music (Sundberg et al., 1982; Friberg, 1991).

6.3 Modeling Expressive Performances

In this section we provide a brief description of the proposed framework. First we describe how expressive dynamics and tempo are encoded. Second, we describe the expectancy and score features. Finally we describe the recurrent neural network (RNN) models used to connect the input features to the expressive targets.

6.3.1 Expressive Parameters

In this section we briefly describe the parameters used to represent expressive tempo and dynamics. For this study, we introduce parameters that describe changes in tempo and loudness. These are in addition to parameters that describe absolute tempo and loudness, which were previously defined in Performance Codec v. 2.0 in Section 4.3.2. Changes in tempo and dynamics were expected to be particularly relevant to the current study, which focused on the role that cognitively-plausible features play in shaping performance. The parameter used to describe tempo in this study also differs from previous chapters in two regards: 1) it represents a smoothed version of the tempo curve and 2) it is not logarithmic. These differences served to make our results more comparable to similar research, in particular (Gingras et al., 2016).

Following the notation from Section 3.2, $\boldsymbol{\mathcal{O}} = \{o_1, \ldots, o_{N_o}\}$ is the set of score onsets (i.e. unique score positions).

1. **Tempo**.
    a) BPR. We take the local beat period ratio as a proxy for musical tempo. In order to compute this parameter, we average the performed onset times of all notes occurring at the same score position and then compute the BPR by taking the slope of the averaged onset times (in seconds) with respect to the score onsets (in beats) and dividing the resulting series by its average beat period. Formally, this can be computed as follows: We define a linear interpolation function of the onset times as

$$\text{onset}_{perf}(\text{onset}(o_i)) = \text{lininterp}(\text{onset}(o_i) \mid \hat{\mathbf{o}}^{perf}, \boldsymbol{\mathcal{O}}), \tag{6.1}$$

where $\hat{\mathbf{o}}^{perf} = \{\hat{o}_1^{perf}, \ldots, \hat{o}_{N_o}^{perf}\}$ is the set of average onset times, which are computed using Equation (3.7). This function has the property of perfect reconstruction, i.e. $\text{onset}_{perf}(\text{onset}(o_i)) = \hat{o}_i^{perf}$. The beat period is then given by

$$\text{BP}(o_i) = \frac{\text{d}}{\text{d}o}\text{onset}_{perf}(o)\bigg|_{o=\text{onset}(o_i)}, \tag{6.2}$$

where the derivative of $\text{onset}_{perf}(o)$ is computed numerically with the finite differences method using a central differences formula (Burden and Faires, 2001). The BPR can be computed as

$$\mathsf{y}_{\text{bpr}}(o_i) = N_o \frac{\text{BP}(o_i)}{\sum_{o_j \in \boldsymbol{\mathcal{O}}} \text{BP}(o_j)}. \tag{6.3}$$

This procedure generates a smoother version of the tempo curve compared to the one defined in Section 3.3.1 that is less affected by spurious effects due to very short performed notes.

b) dBPR. This parameter is computed as the first derivative of BPR with respect to the score position, i.e.

$$\mathsf{y}_{\text{dbpr}}(o_i) = \frac{\text{d}}{\text{d}o}\mathsf{y}_{\text{bpr}}(o)\bigg|_{o=\text{onset}(o_i)}. \tag{6.4}$$

As in the case of BPR, this quantity is computed using numerical differentiation. dBPR corresponds to the *relative acceleration*, i.e. the relative changes in musical tempo.

2. **Dynamics**.

a) VEL. We treat the performed MIDI velocity as a proxy for loudness. This parameter is computed as

$$\mathsf{y}_{\text{vel}}(o_i) = \frac{1}{127}\max \text{vel}(o_i), \tag{6.5}$$

where $\text{vel}(o_i) = \{\text{vel}(n_j) \mid n_j \in o_i\}$ represents the performed MIDI velocity of all notes belonging to o_n.

b) dVEL. This parameter is computed as the first derivative of VEL with respect to the score position, and roughly corresponds to the relative changes in loudness. This parameter is computed as follows: We define a linear interpolation of the values of the MIDI velocity for each onset as

$$\text{vel}_{perf}(\text{onset}(o_i)) = \text{lininterp}(\text{onset}(o_i) \mid \mathbf{vel}, \boldsymbol{\mathcal{O}}), \tag{6.6}$$

where $\mathbf{vel} = \{\mathsf{y}_{\text{vel}}(o_1), \ldots, \mathsf{y}_{\text{vel}}(o_{N_o})\}$ are the values of the normalized MIDI velocity for each onset. As in the case of function onset_{perf}, this function guarantees perfect reconstruction of the performed (normalized) MIDI velocities and generates a smoothed loudness curve.

$$\mathsf{y}_{\text{dvel}}(o_i) = \frac{\text{d}}{\text{d}o}\text{vel}_{perf}(o)\bigg|_{o=\text{onset}(o_i)}. \tag{6.7}$$

This parameter is also computed using numerical differentiation.

6.3.2 Basis Functions

In this section we describe the proposed expectancy and tonal tension features. Since we are interested in studying the contribution of these cognitively motivated features to the predictions of expressive dynamics and tempo, instead of using the full set of basis functions described in Sections 3.3.2, 3.4.1 and 4.2.1, we use a reduced set of basis functions describing low-level information in the score, including some new basis functions that more naturally capture the sequential nature of music, as opposed as the averaging procedure described in Section 4.3.3. We include a full description of the expectancy and tension features in this section, as opposed to the shorter description of basis functions provided in previous chapters, since the definition of these functions is more relevant to the results presented in this chapter.

Expectancy Features (E)

IDyOM provides a conditional probability distribution of a musical event v, given a preceding sequence of events, i.e. $p(v_i \mid v_{i-1}, v_{i-2}, \dots)$. These events are represented as *viewpoints*, following the multiple viewpoint system (Conklin and Witten, 1995). As discussed in Section 3.2.2, the viewpoint representation is not isomorphic to the representation using basis functions. In the following, $v(x_i)$ denotes the viewpoints representing element x_i in the score.

Following the work of Gingras et al. (2016), we use IDyOM to estimate two information-theoretic measures representing musical expectations:

1. **Information content** (IC). The IC measures the unexpectedness of a musical event, and is computed as

$$IC(v(x_i)) = -\log_2 p(v(x_i) \mid v(x_{i-1}), v(x_{i-2}), \dots). \tag{6.8}$$

   a) IC_m. The information content for each melody note. This value is computed using a model that is trained to predict the next chromatic melody pitch using a selection of melodic viewpoints, such as pitch interval (i.e. the arithmetic difference between two consecutive chromatic pitches, measured in MIDI note values), and contour (whether the chromatic pitch sequence rises, falls or remains the same). IDyOM performs a stepwise selection procedure that combines viewpoint models if they minimize model uncertainty as measured by corpus cross entropy (Sears, 2016, pp. 255-275).

$$\varphi_{IC_m}(o_i) = \begin{cases} \frac{1}{Z_{IC_m}} IC(v_{\text{melody}}(n_m)) & \text{if } n_m \in o_i \text{ is a melody note} \\ \varphi_{IC_m}(o_{i-1}) & \text{otherwise} \end{cases} \tag{6.9}$$

   where v_{melody} is the viewpoint representation of the melody notes and Z_{IC_m} is a normalization constant. In the case of Classical piano music, there are fewer contrapuntal textures than in Baroque music, and therefore, we consider only a single main melodic line for each piece (that corresponding to the soprano). We assume therefore that there is at most one melody note per onset. Therefore, if the current onset does not have a melody note, we use the previous value of IC_m. This assumption is musically plausible in scenarios where the notated duration of the melody notes is longer than the duration of the notes in the accompaniment, as is the case of arpeggiated accompaniment figures like the *Alberti bass*, which is common in Classical piano sonatas. For the experiments presented in this chapter, we use the Batik/Mozart dataset, where the main soprano line has been manually annotated. Otherwise, automatic procedures for voice separation, like the one described in McLeod and Steedman (2016), might be required.

Since the values of IC can be potentially be large (in this particular case for the Batik/Mozart dataset they can be an order of magnitude larger that the pitch and metrical features described below), we normalize this feature for a more efficient training of the neural networks models described in Section 6.3.3. In this case, the normalization constant is given as $Z_{IC_m} = \text{mean}(IC_m) + 2\text{std}(IC_m)$, i.e. the mean value of the IC for all melody notes in the pieces of the Batik/Mozart dataset plus two times their standard deviation.

b) IC_c. Estimation of the IC computed for the combination of pitch events (a proxy for harmony) at each score onset. IDyOM predicts the next combination of vertical interval classes above the bass (see Pitch Features 1b below).

$$\varphi_{IC_c}(o_i) = \frac{1}{Z_{IC_c}} IC(v_{\text{chord}}(o_i)). \tag{6.10}$$

where v_{chord} uses vertical interval classes to represent onset o_i and Z_{IC_c} is a normalization constant computed in the same way as Z_{IC_m}.

2. **Entropy** is a measure of the degree of choice or uncertainty associated with a predicted outcome. The entropy can be computed as

$$H(v_i) = \mathbb{E}\{-\log_2 p(v_i \mid v_{i-1}, v_{i-2}, \dots)\} \tag{6.11}$$

a) H_m. Entropy computed for each chromatic pitch in the melody. This is computed in the same fashion as the information content for melody notes, i.e.

$$\varphi_{H_m}(o_i) = \begin{cases} \frac{1}{Z_{H_m}} H(v_{\text{melody}}(n_m)) & \text{if } n_m \in o_i \text{ is a melody note} \\ \varphi_{H_m}(o_{i-1}) & \text{otherwise,} \end{cases} \tag{6.12}$$

where Z_{H_m} is a normalization constant computed in the same way as Z_{IC_m}.

b) H_c. Entropy computed for the combined pitch events at each score onset.

$$\varphi_{H_c}(o_i) = \frac{1}{Z_{H_c}} H(v_{\text{chord}}(o_i)), \tag{6.13}$$

where Z_{H_c} is a normalization constant computed in the same way as Z_{IC_m}.

Tonal Tension Features (T)

In order to characterize tonal tension, we use a set of three quantities computed using the method proposed by Herremans and Chew (2016). These features capture tonal relationships of the music represented in Chew's (2000) *spiral array model*, a three dimensional representation of pitch classes, chords and keys constructed in such a way that spatial proximity represents close tonal relationships. The spiral array is a parametric helix in $\mathbb{R}^3$ represented by

$$\text{sa}(n_i) = \begin{pmatrix} \sin(u(n_i)) \\ \cos(u(n_i)) \\ \frac{\pi}{\sqrt{30}} u(n_i) \end{pmatrix}, \tag{6.14}$$

where $u\colon \boldsymbol{\mathcal{X}} \mapsto \mathbb{R}$ is a function denoting the position of the non-enharmonically equivalent pitch classes (e.g. C♯ and D♭ do not map into the same point) in the circle of fifths given by

$$u(n_i) = \frac{\pi}{2} \text{cof}(n_i), \tag{6.15}$$

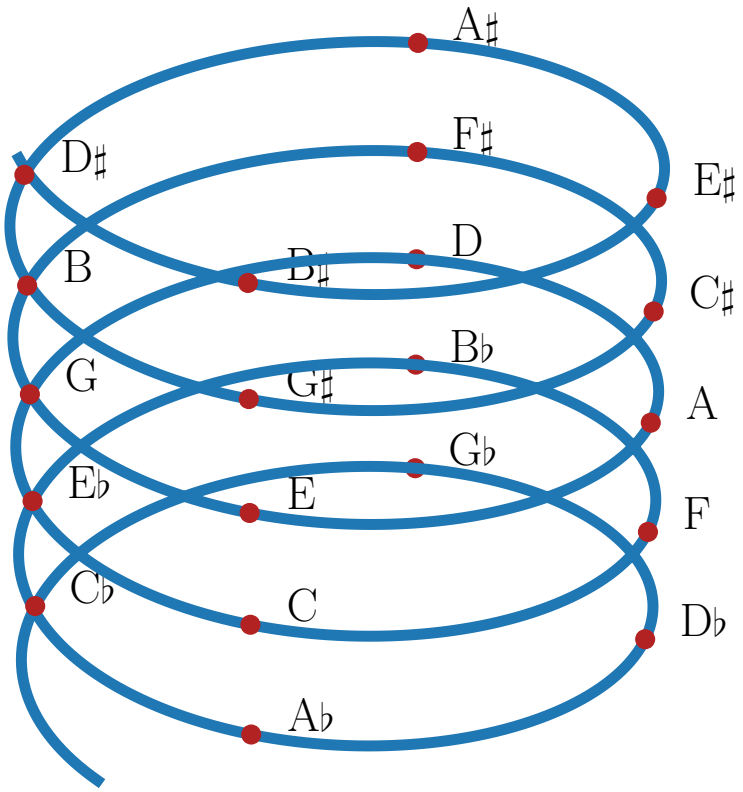


Figure 6.1: Representation of non-enharmonically equivalent pitch classes in the spiral array.

where $\text{cof}(n_i)$ represents the (signed) number of fifths that the pitch class of n_i is away from C such that notes above and below C have positive and negative values, respectively (e.g. $\text{cof}(\text{C}) = 0$, $\text{cof}(\text{B}\flat) = -2$ and $\text{cof}(\text{G}) = 1$)[1]. This geometric location of the notes in the spiral array is shown in Figure 6.1.

Since musical tension varies over time, Herremans and Chew (2016) propose using a sliding window approach. A cloud is defined as the set of points in the spiral array corresponding to one such window. In this work, we define the cloud centered on onset o_i as all active notes within a beat from the center of the cloud, i.e.,

$$\text{cloud}(o_i) = \{\text{sa}(n_j) \mid n_j \text{ is active during } [\text{onset}(o_i) - 1, \text{onset}(o_i) + 1]\}. \tag{6.16}$$

The *center of effect* of a cloud is a point that represents the tonal center of the cloud[2], which can be computed as

$$\text{coe}(\text{cloud}(o_i)) = \frac{\sum_{n_j \in \text{cloud}(o_i)} \text{duration}(n_j) \times \text{sa}(n_j)}{\sum_{n_j \in \text{cloud}(o_i)} \text{duration}(n_j)}. \tag{6.17}$$

The tonal tension features are:

[1] Note that the selection of C as the center of this representation is arbitrary, and a similar representation can be described around any note. The author of this work does not endorse the C-centric hegemony pervasive in Western music.

[2] The tonal center is a concept similar to that of *center of mass* of a dynamical system in classical mechanics.

1. *Cloud diameter* (T_{cd}), which estimates the maximal tonal distance between notes in a segment of music, computed as

$$\varphi_{T_{cd}} = \frac{1}{Z_T} \max \text{pairwisedist}(\text{cloud}(o_i)) \tag{6.18}$$

where pairwisedist is a function that computes the Euclidean distance between all points in the cloud and Z_T is a normalization constant. Since distances in the spiral array can be large (in particular, for the pieces in the Batik/Mozart dataset, these distances can be an order of magnitude larger than the pitch and metrical features described below), we scale all tension features by dividing them by the distance between enharmonically equivalent notes (e.g. C♯ and D♭), which means that $Z_T = \|\text{sa}(\text{C}\sharp) - \text{sa}(\text{D}\flat)\|$.

2. *Cloud momentum* (T_{cm}) quantifies harmonic movement as the tonal distance from a segment of music to the next:

$$\varphi_{T_{cm}}(o_i) = \begin{cases} \frac{1}{Z_T}\|\text{coe}(\text{cloud}(o_i)) - \text{coe}(\text{cloud}(o_{i-1}))\| & i > 0 \\ 0 & i = 0 \end{cases} \tag{6.19}$$

3. *Tensile strain* (T_{ts}), the relative tonal distance between the current segment and the center of effect of the key of the piece.

$$\varphi_{T_{ts}}(o_i) = \frac{1}{Z_T}\|\text{coe}(\text{cloud}(o_i)) - \text{coe}_{\text{key}}(o_i))\|, \tag{6.20}$$

where $\text{coe}_{\text{key}}(o_i)$ is the center of effect of a key, i.e. a point in the spiral array model that represents the tonal center of a key. More specifically, the center of effect of a key is computed as a linear combination of the center of effect of its tonic, dominant and subdominant chords (see Section 3.4 in (Chew, 2000)).

Herremans and Chew (2016) evaluated these features by comparing them to the empirical study by Farbood (2012), showing that these features correlate to human perception of tonal tension.

Pitch and Metrical Features

We include low-level descriptors of the musical score that have been shown to predict characteristics of expressive performance.

1. **Pitch (P)**

   a) $(pitch_h, pitch_l, pitch_m)$. Three features representing the chromatic pitch of the highest note, the lowest note, and the melody note at each onset.

   b) (vic_1, vic_2, vic_3). Three features describing up to three vertical interval classes above the bass, i.e. the intervals between the notes of a chord and the lowest pitch, excluding pitch class repetition and octaves. For example, a C major triad (C, E, G), starting at C_4 would be represented as $(\ pitch_l \quad vic_1 \quad vic_2 \quad vic_3\ ) = (\ \frac{60}{127} \quad \frac{4}{11} \quad \frac{7}{11} \quad 0\ )$, where 0 denotes the absence of a third interval above C_4, i.e. the absence of a fourth note in the chord.

2. **Metrical position (M)**

   a) b_ϕ. This feature, referred to as the *beat phase* (Xia et al., 2015), represents the relative location of an onset within the bar.

b) (b_d, b_s, b_w). Three binary features encoding the metrical strength of the t-th onset. b_d is nonzero at the downbeat; b_s is nonzero at the secondary strong beat in duple meters (e.g. quarter-note 3 in $\frac{4}{4}$, and eighth-note 4 in $\frac{6}{8}$), and b_w is nonzero at weak metrical positions (i.e. whenever b_d and b_s are both zero).

A mathematical definition of these features is provided in Appendix A.1

6.3.3 Computational Model

As described in Section 4.3, As described in Chapter 4, we can use recurrent neural networks (RNNs), a family of non-linear sequential models, to assess the contribution of the features described above to the prediction of expressive dynamics and tempo. In this work, we use a simple architecture, which we will refer to as bRNN, consisting of a composite bidirectional long short-term memory layer (LSTM) with multiplicative integration (Wu et al., 2016) with 10 units (5 units processing information forwards and 5 processing information backwards) and a linear dense layer with a single unit as output. In this way, we can model the each expressive parameter (BPR, dBPR, VEL or dVEL) as

$$f(\boldsymbol{\varphi}(o_i)) = \mathbf{w}_h^{\mathrm{T}} \begin{pmatrix} \boldsymbol{h}_{i;\mathrm{fw}}^{(l)}(\boldsymbol{\varphi}(o_i) \mid \theta_f) \\ \boldsymbol{h}_{i;\mathrm{bw}}^{(l)}(\boldsymbol{\varphi}(o_i) \mid \theta_b) \end{pmatrix} + \mathbf{w}_x^{\mathrm{T}} \boldsymbol{\varphi}(o_i) + w_0, \tag{6.21}$$

where $\boldsymbol{h}_{i;\mathrm{fw}} \in \mathbb{R}^5$ and $\boldsymbol{h}_{i;\mathrm{bw}} \in \mathbb{R}^5$ are the forwards and backwards LSTM layers with parameters θ_f and θ_b, respectively; $\mathbf{w}_h \in \mathbb{R}^{10}$ and $\mathbf{w}_x \in \mathbb{R}^M$ are weight vectors connecting the hidden layers and the input features to the output of the network, respectively, where M is the number of basis functions in the feature set; and $w_0 \in \mathbb{R}$ is a scalar bias. The bidirectional layer used in this architecture concatenates the forward and backward components of the recurrent layer, as described in Equation (4.31). The main difference between the RNN architecture described in this section and the one discussed in Section 4.3.5 is the use of the more sophisticated LSTMs with multiplicative integration, compared to the vanilla recurrent layers[3]. Appendix F.2.2 contains a detailed description of the LSTMs with multiplicative integration.

6.4 Experiments

For each expressive parameter, we perform 18 5-fold cross-validation experiments corresponding to models trained on the feature sets described in Section 6.3.2: pitch features (denoted as **P**), metrical features (denoted as **M**), expectancy features (denoted as **E**) and tonal tension features (denoted as **T**), and all combinations thereof (e.g. **P** + **M** denotes a feature set consisting of pitch and metrical features, or **E** + **T** denotes a feature set consisting of expectancy and tonal tension features). Additionally, we include two feature sets consisting of a selection of features using a feature selection (FS) method described below (**P** + **M** +**T** (FS) and **P** + **M** + **E** + **T** (FS)). All feature sets are listed in the first column of Table 6.1. The 5-fold cross-validation experiment are conducted as follows: each model is trained/tested on 5 different partitions (folds) of the dataset, which is organized into training and test sets, such that each piece in the corpus occurs exactly one in the test set. For each fold, we use 80% of the pieces for training and 20% for testing the model. The training set was further split into a set for updating the parameters of the network with 80% of the pieces in the training set and a validation set with

[3] Although originally published in early 2017 in (Grachten and Cancino-Chacón, 2017), the experimental results described in Section 4.3.5 were conducted in late 2015/early 2016, whereas the article that introduced the LSTM with multiplicative integration was published in late 2016 (Wu et al., 2016).

Feature Set	Tempo				Dynamics			
	BPR		dBPR		VEL		dVEL	
	R^2	r	R^2	r	R^2	r	R^2	r
P	0.024	0.164	0.068	0.262	0.326	0.586	0.236	0.486
M	0.051	0.241	0.093	0.305	0.048	0.242	0.041	0.195
P + M	0.056	0.254	0.105	0.325	0.351	0.615	0.250	0.501
E	0.028	0.179	0.030	0.174	0.054	0.238	0.024	0.158
P + E	0.035	0.196	0.075	0.275	0.331	0.590	0.231	0.481
M + E	0.067	0.268	0.100	0.318	0.072	0.281	0.054	0.224
P + M + E	0.060	0.263	0.113	0.334	0.355	0.617	0.250	0.501
P + M + E (FS)	0.051	0.248	0.106	0.326	0.332	0.590	0.251	0.503
T	0.010	0.120	0.011	0.110	0.018	0.157	0.026	0.163
P + T	0.021	0.155	0.073	0.271	0.335	0.597	0.250	0.500
M + T	0.054	0.245	0.092	0.305	0.052	0.250	0.050	0.214
P + M + T	0.054	0.246	0.110	0.331	0.347	0.611	0.282	0.532
P + M + T (FS)	0.053	0.241	0.104	0.325	0.280	0.556	0.257	0.508
E + T	0.039	0.209	0.035	0.191	0.035	0.203	0.030	0.171
P + E + T	0.045	0.220	0.079	0.281	0.336	0.599	0.249	0.499
M + E + T	0.061	0.265	0.099	0.317	0.074	0.290	0.064	0.252
P + M + E + T	0.064	0.269	0.115	0.337	0.362	0.623	0.255	0.505
P + M + E + T (FS)	0.052	0.238	0.099	0.318	0.270	0.540	0.242	0.493

Table 6.1: Predictive results for expressive tempo and dynamics, averaged over all pieces on the Batik/Mozart corpus. Larger R^2 and r means more accurate predictions.

20% of the pieces. The parameters of the model are learned by minimizing the mean squared error on the training set using RMSProp (Tieleman and Hinton, 2012).

For reproducibility, the hyper-parameters for training the models are given as follows: The weights of the bidirectional LSTM with multiplicative integration were drawn from $\mathcal{U}(-0.01, 0.01)$, and the biases and initial states were initialized to zero. We used a learning rate of 10^{-3}, a gradient moving average decay factor of 0.9 for computing the parameter updates with RMSProp. The value of the gradients was clipped to lie between -2 and 2. The gradients for back-propagation were truncated after 100 steps. The batch size was set to 100 sequences of 200 steps each. To avoid overfitting, the l_2-norm weight regularization was used with regularization coefficient of 10^{-3}. Early stopping was used after 100 epochs without improvement in the validation set. All models were trained for a maximum of 5000 epochs.

The feature selection procedure computes the pairwise *mutual information* between each of the features and each of the expressive parameters. This information-theoretic measure expresses how much knowing the value of one variable reduces uncertainty about the value of the other variable (Ross, 2014), and is a common way of determining the relevance of features in prediction tasks. Formally, the mutual information between two variables a and b is given by

$$\mathrm{MI}(a, b) = \mathbb{E}\left\{\log\left(\frac{p(a, b)}{p(a)p(b)}\right)\right\},$$

where $\mathbb{E}$ is the expectation operator, $p(a, b)$ is the joint probability distribution of a and b, and $p(a), p(b)$ are the marginal probability distributions of a and b, respectively. If a and b are statistically independent (i.e. they do not share information about each other), the mutual information is zero. In the FS scenario we select the 10 features with the largest mutual information for each expressive parameter. This procedure was performed on a small subset of the Batik/Mozart dataset (20% of the pieces selected randomly).

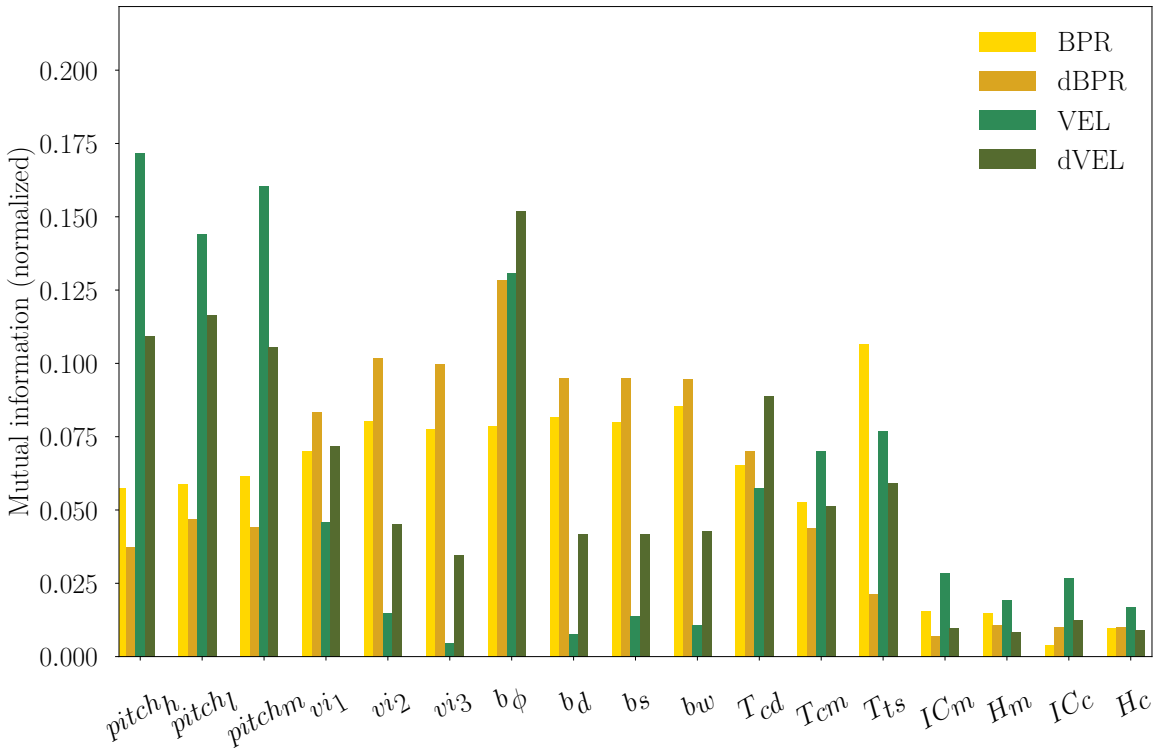


Figure 6.2: Normalized mutual information between features and expressive parameters. A larger mutual information means that the variables are more related to each other.

6.5 Results and Discussion

6.5.1 Predictive accuracy

We use the coefficient of determination R^2 and Pearson's correlation coefficient r to evaluate model accuracy. Table 6.1 shows the values R^2 and r averaged over all pieces on the Batik/Mozart corpus. For each expressive parameter, we conducted a paired–samples two-tailed t-test at the $p < 0.01$ level to compare the differences between the R^2 values of features sets with and without expectancy features, tension features or both groups. (e.g. **P** vs. **P** + **T**,**M** vs. **M** + **T**, and **M** + **P** vs. **M** + **P** + **T** for tension features and **P** vs. **P** + **E**, **M** vs. **M** + **E**, and **M** + **P** vs. **M** + **P** + **E** for expectancy features). The results of these comparisons are reported in Table 6.2 for parameters describing expressive tempo and Table 6.3 for parameters describing expressive dynamics.

Figure 6.2 shows the normalized mutual information between each feature and the expressive parameters. For the benefit of graphical display (due to differences in ranges of mutual information), a normalization was performed by dividing the value of the mutual information between each feature and a given expressive parameter by the sum of the mutual information for all features for that parameter. In this plot, the height of a column (the value of the mutual information) signifies how closely related that feature is to the expressive parameter. The results in this plot suggest that the tension features, in particular the cloud diameter might be more related to the prediction of changes in tempo and dynamics, whereas the tensile strain might be more related to absolute tempo and dynamics than their changes.

Note that the values in Figure 6.2 only measure the MI of the features and expressive parameters

	Tempo					
	BPR			dBPR		
Feature Set		+ **E**	*d*		+ **E**	*d*
∅	-	0.028		-	0.030	
P	0.024	**0.035**	(0.36)	0.068	**0.075**	(0.14)
M	0.051	**0.067**	(0.27)	0.093	**0.100**	(0.12)
P + **M**	0.056	0.060		0.105	**0.113**	(0.11)
P + **M** + **T**	0.054	**0.064**	(0.16)	0.110	**0.115**	(0.06)
		+ **T**	*d*		+ **T**	*d*
∅	-	0.010		-	0.011	
P	0.024	0.021		0.068	**0.073**	(0.10)
M	0.051	0.054		0.093	0.092	
P + **M**	0.056	0.054		0.105	**0.110**	(0.06)
P + **M** + **E**	0.060	0.064		0.113	0.115	
		+ **E** + **T**	*d*		+ **E** + **T**	*d*
∅	-	0.039		-	0.035	
P + **M**	0.056	0.064		0.105	**0.115**	(0.12)

Table 6.2: Proportion of variance explained (R^2) for expressive tempo using different feature sets, averaged over all pieces on the Batik/Mozart corpus. Larger R^2 values reflect more accurate predictions. For each combination of target and feature set, the results are listed for that feature set as is (left), and including expectancy features **E**, tonal tension features **T** or both **E** + **T** (right). For clarity, improvements of R^2 as a result of adding **E**, **T** or both are marked in green, whereas deteriorations are marked in red. Bold numbers mark a statistically significant difference ($p < 0.01$). The effect size (Cohen's d) is reported in parenthesis for those cases with statistically significant differences.

at isolated time instances, without context. Although this gives a good first impression of the relevance of features, the bRNN model presented above is specifically designed to take advantage of the temporal context to make predictions, implying that feature values at times before and after τ may also help to predict expressive parameters at τ. Therefore Figure 6.2 does not necessarily reflect the relevance of features when used as input to the bRNN, which is reflected in the results for (FS) feature sets in Table 6.1 being consistently worse than their counterparts without feature selection.

The results in Tables 6.2 and 6.3 show that the effects of including **E** were most significant for prediction of BPR and dBPR, but (mostly) not significant for predicting VEL and dVEL, suggesting that models including expectancy features are better for predicting expressive tempo, particularly changes in tempo. The fact that the use of expectancy features improves model performance for tempo but not for dynamics might be due to the relation of expressive tempo to structural properties of the music, such as phrase-final lengthening and the final *ritardando* at the end of a piece (Honing, 2006). Since expectation features also relate to music structure in the sense that music tends to be more unpredictable at boundaries between musical segments than within segments (Pearce et al., 2008), this may in part explain why the models are better at predicting changes in expressive tempo (dBPR).

On the other hand, according to the results in Tables 6.2 and 6.3, the effect of including **T** was

	Dynamics					
	VEL			dVEL		
Feature Set		+ **E**	d		+ **E**	d
$\emptyset$	-	0.054		-	0.024	
P	0.326	0.331		0.236	0.231	
M	0.048	**0.072**	0.62	0.041	**0.054**	0.27
P + **M**	0.351	0.355		0.250	0.250	
P + **M** + **T**	0.347	**0.362**	0.12	**0.282**	0.255	0.32
		+ **T**	d		+ **T**	d
$\emptyset$	-	0.018		-	0.026	
P	0.326	0.335		0.236	**0.250**	0.16
M	0.048	0.052		0.041	**0.050**	0.18
P + **M**	0.351	0.347		0.250	**0.282**	0.40
P + **M** + **E**	0.355	0.362		0.250	0.255	
		+ **E** + **T**	d		+ **E** + **T**	d
$\emptyset$	-	0.035		-	0.030	
P + **M**	0.351	0.362		0.250	0.255	

Table 6.3: Proportion of variance explained (R^2) for expressive dynamics using different feature sets, averaged over all pieces on the Batik/Mozart corpus. Larger R^2 values reflect more accurate predictions. For each combination of target and feature set, the results are listed for that feature set as is (left), and including expectancy features **E**, tonal tension features **T** or both **E** + **T** (right). For clarity, improvements of R^2 as a result of adding **E**, **T** or both are marked in green, whereas deteriorations are marked in red. Bold numbers mark a statistically significant difference ($p < 0.01$). The effect size (Cohen's d) is reported in parenthesis for those cases with statistically significant differences.

not significant for the prediction of BPR and VEL but it *was* significant for the prediction of dVEL in all cases. For dBPR including **T** was only beneficial in combination with **P**. Based on this result we hypothesize that the concept of tonal tension is relevant for changes in tempo, but the features used to represent tonal tension may not convey enough information by themselves and are therefore only advantageous in combination with more specific pitch information.

Furthermore, the results in Tables 6.2 and 6.3 suggest that using both expectancy and tension features does not necessarily improve prediction of the expressive parameters, which supports the hypothesis that different expressive parameters might need different sets of features, as discussed in Section 3.6.

6.5.2 Sensitivity Analysis

In order to visualize the contribution of each feature to the prediction of the changes in tempo and loudness, we perform a differential sensitivity analysis[4] of the models by computing a local linear approximation of the output of the bRNNs trained on all features (**P** + **M** + **E** + **T**). The sensitivity of the model outputs to the input features for each expressive parameter is computed

[4]We use the definition of sensitivity analysis from the applied mathematics literature, not in the sense used in the psychology literature.

as

$$S_{i,j} = \mathbb{E}\left\{\frac{\partial}{\partial\varphi_i(o_j)} f(\boldsymbol{\varphi}(o_j)))\right\} \approx \frac{1}{|\mathcal{Q}|}\sum_{\varphi_i(o_j^l)\in\mathcal{Q}} \frac{\partial}{\partial\varphi_i(o_j^l)} f(\boldsymbol{\varphi}(o_j^l)), \tag{6.22}$$

where $\mathcal{Q} = \{\{\varphi(o_1^1),\ldots,\varphi(o_{N_l}^1)\},\ldots,\{\varphi(o_1^{N_q}),\ldots,\varphi(o_{N_l}^{N_q})\}\}$ is a set of input sequences selected randomly from the dataset. The resulting sensitivity maps are plotted in Figure 6.3. For these plots, 2500 sequences were randomly selected from the pieces in the Batik/Mozart dataset[5]. Each sequence contains 400 time steps, corresponding to $\tau-199,\ldots,\tau-1,\tau,\tau+1,\ldots,\tau+200$ (although for the plots we focus on $\tau-5$ to $\tau+5$). This figure can by roughly interpreted as follows: The color in the cell that corresponds to feature φ_i and time step j represents the (average) contribution of the value of φ_i at time j (i.e. $\varphi_i(o_j)$) to the prediction of the expressive parameter at time $j=\tau$ (the center column of the plot). Blue tones reflect feature values that negatively contribute to the predicted value (the higher the feature value the lower the predicted value), and red tones reflect feature values that positively contribute to the predicted value.

The plots in Figure 6.3 suggest a tendency of the performer to emphasize certain melodic and harmonic events by slowing down. For example, there is a tendency to slow down if the next melodic events are unexpected (see the reddish hue in IC_m for time steps $>\tau$ in the top right plot) and to speed up if the previous melodic events were unexpected or uncertain (the blueish hue in H_m and IC_m for time steps $\leq\tau$ in the top right plot), which is consistent with the findings reported in (Gingras et al., 2016). On the other hand, a passage consisting of uncertain or unexpected harmonic events contributes to an overall slower tempo (the reddish hue in H_c and IC_c for time steps $\geq\tau$ in the top left plot).

Regarding tonal tension, according to the sensitivity maps in Figure 6.3, tonally distant melody notes – which contribute to increased tension, captured by an increase in cloud diameter (T_{cd}) – might be emphasized by slowing down (the reddish hue in T_{cd} for time step τ in the top right plot). Furthermore, the sensitivity maps suggest that such a slowing down before tonally distant melodic events might be prepared for with a slight speeding up (the bluish hue in T_{cd} for time steps $>\tau$ in the top right plot). This finding is consistent with the melodic charge rule in the KTH model (Friberg et al., 2006). On the other hand, harmonic events that contribute to the increase of tension – such as chords that are tonally far away from the key of the piece (as is the case of the more unstable parts of the development section of a sonata form), captured by the increase in tensile strain (T_{ts}), or consecutive chords that are tonally far away from each other, captured by the cloud momentum (T_{cm}) – are emphasized by slowing down and an increase in loudness (the reddish hue for time step τ in T_{cm} in the top right plot and in T_{ts} in the bottom right plot, respectively). This finding agrees with the harmonic charge rule in the KTH model.

6.6 Conclusions

In this chapter we have investigated the plausible role that musical expectations and tonal tension may play in shaping musical expression in classical piano performances. Our results support the view that expectancy features, reflecting what a listener is expecting to hear, can

[5] As in the case of the sensitive-difference graphs described in 5.4, we transpose the sensitivity maps such that the temporal dimension is in the x axis.

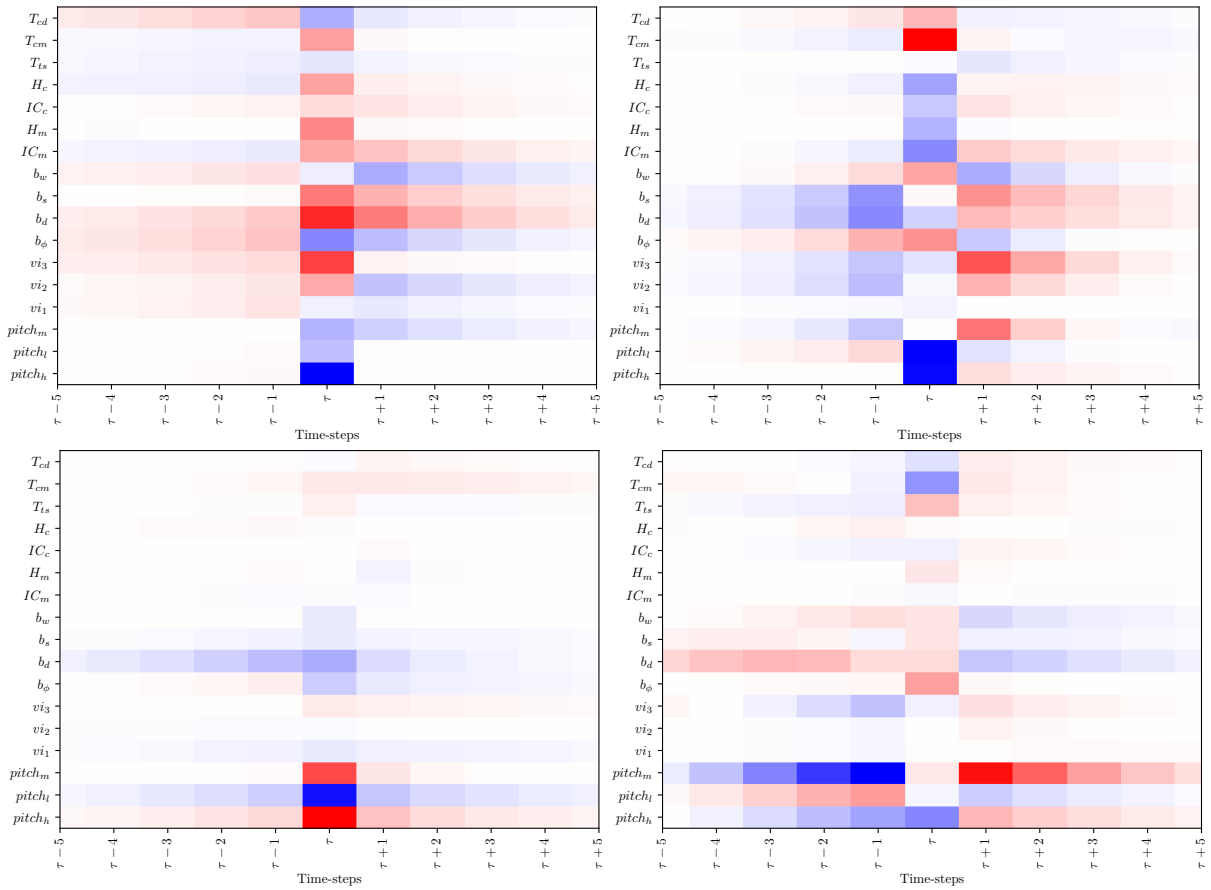


Figure 6.3: Sensitivity plots for BPR (top left), dBPR (top right), VEL (bottom left) and dVEL (bottom right). Each row in the plot corresponds to an input feature and each column to the contribution of its value at that time step to the output of the model at τ (the center of each plot). Red and blue indicate a positive and negative contribution, respectively.

help to predict the way pianists perform a piece. In particular, our results show that using expectancy features improves predictions of tempo, particularly changes in tempo, but does not improve predictions of dynamics. On the other hand, our results also show that using tonal tension information improves predictions of change in dynamics, but not predictions of absolute tempo and dynamics. For predicting changes in tempo, using tonal tension features as defined in (Herremans and Chew, 2016) was only beneficial when low level pitch information was also available. This suggests that tonal tension features are potentially relevant for predicting tempo changes, but by themselves not sufficiently specific for that purpose.

The sensitivity analysis of the trained models found some evidence relating well-known rules/guidelines for performance, as codified by the rules in the KTH model (Friberg et al., 2006) and the results by Gingras et al. (2016).

Future work may focus on a more explicit testing of the hypothesis that recurrent neural network models may learn features describing tonal characteristics from low level pitch information as a side effect of learning to predict expressive tempo and dynamics.

7 Implementing the Basis Function Models: The Basis Mixer

This chapter contains material published in

- Cancino Chacón, C. E. and Grachten, M. (2016). The Basis Mixer: A Computational Romantic Pianist. In *Late Breaking/ Demo, 17th International Society for Music Information Retrieval Conference (ISMIR 2016)*, New York, NY, USA

7.1 Introduction

This chapter discusses how to implement the BM framework as a standalone system for generating an expressive performance of a piece given its score. This system, which we call the *Basis Mixer* allows a user to render an expressive performance of a piece given its score in MusicXML format. Additionally, we present a user interface by means of a web app that combines the Basis Mixer with a web service. Besides rendering of expressive performances, By manipulating individual parameters to varying degrees, the Basis Mixer could be used as an educational tool for teaching about expressive variations (e.g. contrasting metronomic with highly temporally varied performances), playing styles (e.g. showing models trained on different performers), the modeling process (e.g. using different versions of the predictive models – for example linear models vs. FFNNs), etc.

The BM framework for musical expression was first proposed by Grachten and Widmer (2012). As described in Chapter 3, this framework uses numerical encodings of musical scores (referred to as basis functions) to learn how music is performed in an expressive manner. Chapters 3, 4, 5, 6 discuss the use of the BM framework for both analysis and synthesis of music performances.

As discussed in Section 3.2, the BM consists of three components:

1. *A performance codec* $\mathcal{C}$, which denotes the way the expressive characteristics of a performance are represented by the model through the so-called *expressive parameters*.
2. *A score representation model* $\mathcal{Y}$, which describes the way the information in a musical score is represented. In the particular case of the BM framework, we use *basis functions* to capture certain structural aspects of the score and represent them numerically.
3. *A predictive function* F, which maps elements of the score represented through the basis functions to the expressive parameters.

Initial versions of the framework, described in Chapter 3, were limited to simple linear models to describe the relation between expressive dynamics and the musical score in classical piano music (Grachten and Widmer, 2012; Grachten et al., 2014), Subsequent versions include the use of non-linear models, like the use of feed-forward neural networks (FFNNs) (Cancino Chacón and Grachten, 2015; Cancino-Chacón et al., 2017d), and recurrent neural networks (RNNs) (Grachten and Cancino-Chacón, 2017; Cancino-Chacón et al., 2017c; Cancino-Chacón

and Grachten, 2018). These non-linear and sequential models are described in more detail in Chapter 4.

In addition to expressive dynamics and tempo, the BM framework represents articulation and timing deviations of individual notes, and thus, allows for *complete* representation of expressive piano performances (in MIDI format).

The contributions of this chapter are twofold: First, we present a description of the implementation aspects of each of the components of a BM framework, and how this implementation can be used for rendering an expressive performance of a piece given its score (in MusicXML), as well as describing how to train the models given a set of performances matched to their scores. Secondly, we present an interactive web interface that allow users to control some aspects of the expressive performance generated by the Basis Mixer.

The code of the Basis Mixer will be made publicly available[1]. Note that due to the ongoing nature of the development of the BM framework, the latest state of the Basis Mixer in the repository might not correspond to the one described in this chapter.

Due to copyright reasons, we are not allowed to distribute the datasets used for training the models (the Magaloff/Chopin and Beethoven/Zeilinger datasets). Instead, we provide several pre-trained models corresponding to different model architectures/composer combinations.

The Basis Mixer is implemented in Python 2.7[2], an object-oriented programming language, and relies heavily on Numpy[3], the main Python package for scientific computing.

The rest of this chapter is structured as follows: Section 7.3 describes the implementation the performance codec. In Section 7.4 we give an overview of the way a musical score (in MusicXML format) is represented using the BM framework. Section 7.5 describes the implementation of the predictive functions. In Section 7.6 we present the prototype for a web service that allows the user to generate and manipulate an expressive performance of a piece of music using BM models, given a MusicXML file. Finally, conclusions are presented in Section 7.7.

7.2 Overview of the Basis Mixer

In this section we provide a detailed overview of the way the different components of the Basis Mixer interact with each other to generate a rendering of a piece given its score, and to train the predictive functions given a training set.

The purpose of this section is to provide a guideline and motivation for each of the components of the system, which will be described in the following sections. In the following overview, elements denoted with the typewriter typeface (e.g. `PerformanceCodec`, `RegressionModel`, etc.) correspond to components of the Basis Mixer.

7.2.1 Rendering a Performance

Figure 7.1 illustrates the process of rendering the performance of a piece given its score using the Basis Mixer. The black arrows in this figure denote the flow of information between the different components. The blue arrows denote sharing of information necessary to initialize a

[1] https://github.com/OFAI/basismixer.
[2] https://www.python.org.
[3] http://www.numpy.org

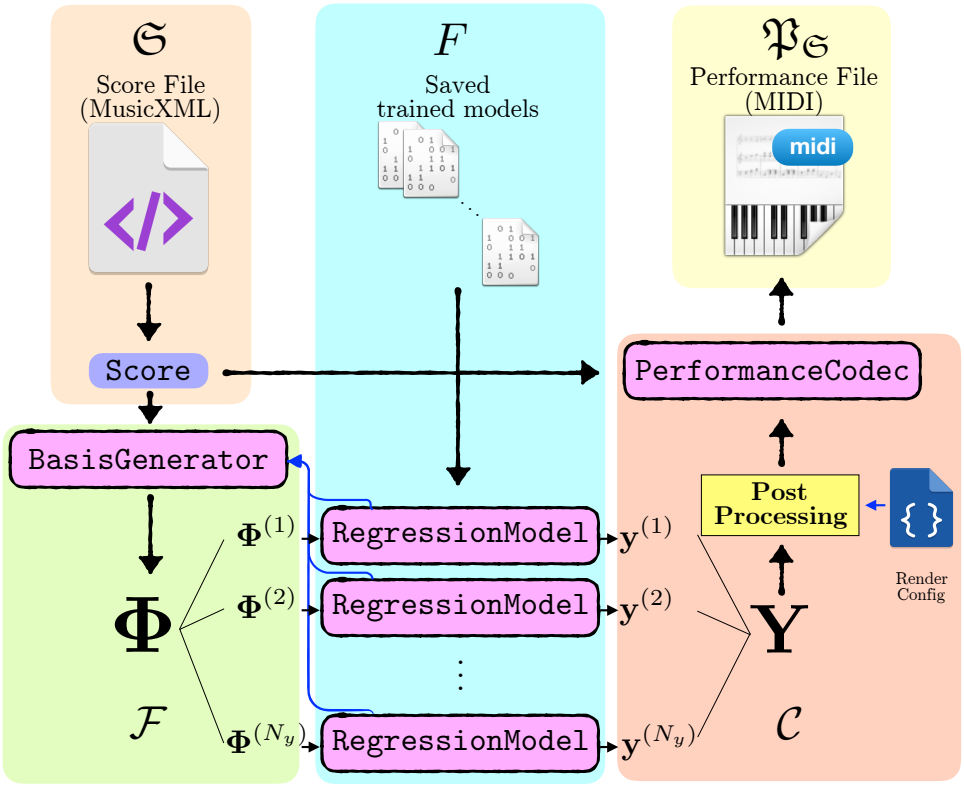


Figure 7.1: Schematic representation of the relationship between components of the Basis Mixer for rendering the performance of a piece given a score $\mathfrak{S}$. Pink rectangles with rounded corners and thick black borders represent instances of the main components of the system: `BasisGenerator`, `RegressionModel`, and `PerformanceCodec`. The Yellow rectangle with a thin black border represents a method for post-processing the predicted performance $\mathbf{Y}$: rescale and recenter and use the specified average BP. Black arrows denote the direction of the flow of information. Blue arrows denote sharing of configuration information passed to the constructor of the instances of the main components. The background colored rectangles without a border encapsulate the components of the Basis Mixer that correspond to the main components of the BM framework: the score $\mathfrak{S}$ (in orange), the performance $\mathfrak{P}_{\mathfrak{S}}$ (in light yellow), the score representation model $\mathcal{F}$ (in green), the performance codec $\mathcal{C}$ (in red), and the predictive function F (in blue).

component of the Basis Mixer. Furthermore, the relationship of the components of the Basis Mixer and the BM framework is highlighted with the background colored rectangles.

The Basis Mixer renders the performance of a piece is as follows:

Initializing the Basis Mixer

1. Initialize an instance of class `PerformanceCodec`, which implements a performance codec $\mathcal{C}$. This object defines the version of the expressive parameters used in describing the performance. In Figure 7.1, the `PerformanceCodec` object defines N_y parameters. The `PerformanceCodec` is described in more detail in Section 7.3 below.

2. Load the trained predictive functions ($f(\cdot)$) for the parameters specified in the instance of `PerformanceCodec`. Each of these models is implemented as an instance of class

`RegressionModel`, which is described in Section 7.5 below. The whole set of functions specifies the complete predictive function F. If no function for an expressive parameter is provided, the system will use a dummy function which returns the parameters equivalent to a deadpan performance. In Figure 7.1, there are N_y `RegressionModel` objects, implementing functions $f^{(1)}, \ldots, f^{(N_y)}$, respectively, each of which corresponds to an expressive parameter.

3. Initialize an instance of `BasisGenerator`, a class which implements a score representation model ($\mathcal{Y}$), which defines the basis functions used for modeling each expressive parameter. These basis functions are specified by the trained predictive functions (since a predictive function cannot predict an expressive parameter for a feature that was never seen in the training set). As denoted in Figure 7.1, there is a single `BasisGenerator` object, which generates the input matrices of score representations for each predictive function. The `BasisGenerator` is described in Section 7.4 below.

Generating the Performance

1. Parse the score in MusicXML into a `Score` object (described in Section 7.4 below) and compute the performance representation matrices $\mathbf{\Phi}$ (implemented as Numpy arrays) corresponding to the input for each expressive function. In Figure 7.1, these matrices are $\mathbf{\Phi}^{(1)}, \ldots, \mathbf{\Phi}^{(N_y)}$, which correspond to each of the `RegressionModel` objects.

2. Compute the predictions of each predictive function $f(\mathbf{\Phi}) = \mathbf{y}$ using the `RegressionModel` objects. In Figure 7.1, these predictions are $\mathbf{y}^{(1)}, \ldots, \mathbf{y}^{(N_y)}$, which are then concatenated into a single $\mathbf{Y}$ (see discussion of the form of $\mathbf{y}$ for the case of onset-wise and note-wise parameters in Section 4.3.4).

3. Post-process the predictions of the model: the predictions of each parameter are standardized to have zero-mean and unit variance, and then are rescaled and recentered according to a user-specified configuration file (in the current implementation, a JSON file[4]). This post-processing module (including the configuration file) is depicted in Figure 7.1. The purpose of the post-processing is to allow the user to control aspects such as the average tempo and maximal loudness of the performance.

4. Use the `PerformanceCodec` object to decode the predicted performance into a MIDI file using the score information in the `Score` object (see the arrow connecting `Score` to `PerformanceCodec` in Figure 7.1).

Audio examples of pieces rendered by the Basis Mixer are available online[5].

7.2.2 Training the Basis Mixer

As described above, in order to render a performance the Basis Mixer requires models trained on a dataset of performances matched to their scores. This process is illustrated in Figure 7.2, which follows the same color and shape conventions as Figure 7.1.

The training of the Basis Mixer is described as follows:

1. Select the version of the performance codec $\mathcal{C}$ and initialize its corresponding `PerformanceCodec` object.

[4] JavaScript Object Notation.
[5] http://www.carloscancinochacon.com/documents/online_extras/basis_mixer/basis_mixer.html.

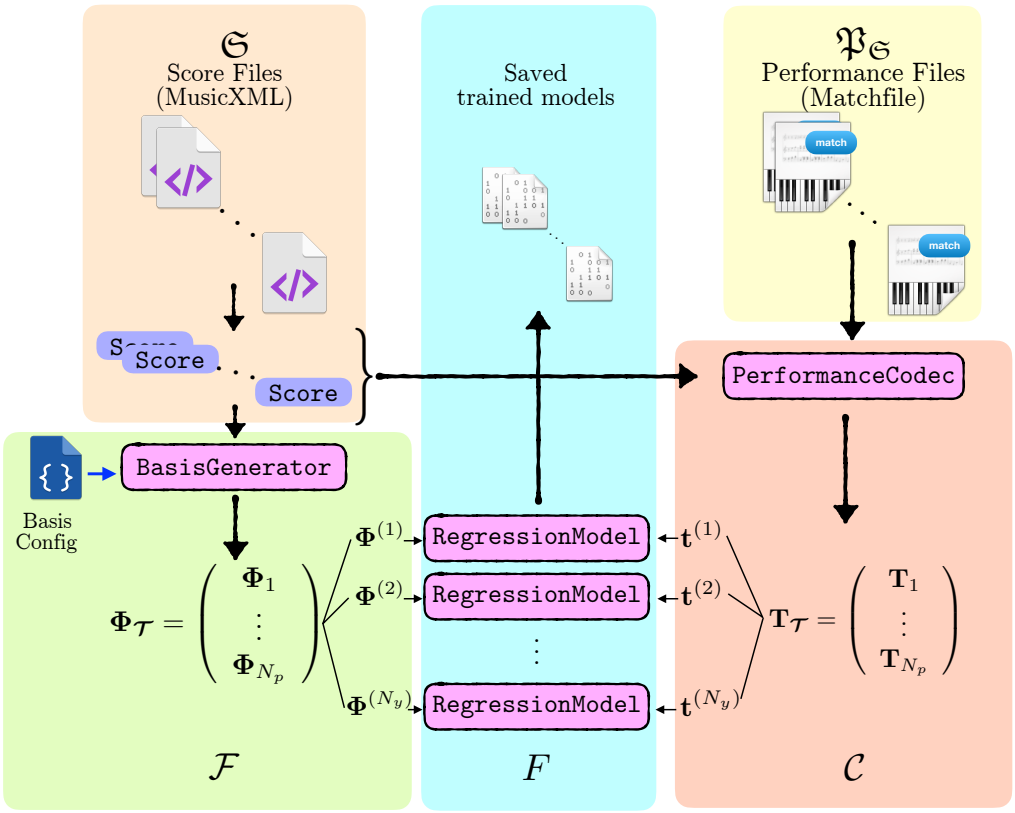


Figure 7.2: Schematic representation of the relationship between components of the Basis Mixer for training the predictive function given a test set $\mathcal{T}$. Pink rectangles with rounded corners and thick black borders represent instances of the main components of the system: `BasisGenerator`, `RegressionModel`, and `PerformanceCodec`. Black arrows denote the direction of the flow of information. Blue arrows denote flow of configuration information passed to the constructor of the instances of the main components. The background colored rectangles without a border encapsulate the components of the Basis Mixer that correspond to the main components of the BM framework: the score $\mathfrak{S}$ (in orange), the performance $\mathfrak{P}_{\mathfrak{S}}$ (in light yellow), the score representation model $\mathcal{F}$ (in green), the performance codec $\mathcal{C}$ (in red), and the predictive function F (in light blue).

2. Define the performance representation model $\mathcal{Y}$ and initialize its corresponding `BasisGenerator` by specifying the basis functions used for modeling each expressive parameter in the performance codec. In the current implementation this specification is done through a user provided configuration file in JSON format (see the "Basis Config" providing initialization settings to `BasisGenerator` (denoted by a blue arrow) in Figure 7.2).

3. Given N_p matched performances (in Matchfile format, described in Section 7.3 below) and their corresponding scores (in MusicXML format), construct the training dataset as follows:

   a) Parse each score in the dataset into a `Score` object

   b) Use these `Score` objects to generate the score representations corresponding to each of the pieces using the `BasisGenerator`. In Figure 7.2, the scores in the training set, denoted as $\mathbf{\Phi}_{\mathcal{T}}$, are represented by a matrix concatenating all matrices corresponding to each piece in the training set (see the discussion regarding the form of

the performance representation for onset-wise and note-wise parameters in Section 4.3.3).

c) Encode each matched performance using the `PerformanceCodec` (and the `Score` object corresponding to the score of the performance). The performance representation (in terms of the expressive parameters defined by the performance codec $\mathcal{C}$) of the pieces in the training set is denoted as $\mathbf{T}_{\boldsymbol{\mathcal{T}}}$. In Figure 7.2, $\mathbf{T}_{\boldsymbol{\mathcal{T}}}$ is a matrix concatenating the performance representations of each piece in the dataset. This this performance representation is split into vectors $\mathbf{t}$, each denoting the components of $\mathbf{T}_{\boldsymbol{\mathcal{T}}}$ corresponding to an expressive parameter (see $\mathbf{t}^{(1)}, \ldots, \mathbf{t}^{(N_y)}$ in Figure 7.2). We remind the reader that the difference between $\mathbf{y}$ in Figure 7.1 and $\mathbf{t}$ in Figure 7.2 is that $\mathbf{y}$ are predictions of the model, whereas $\mathbf{t}$ represent actual observed performances.

4. Specify the architecture of the predictive functions $f(\cdot)$ predicting the expressive parameters and the hyper-parameters for its optimization. In the current implementation this is done through a configuration files. Each of these predictive functions is represented with a `RegressionModel`.

5. Use the training set to train each `RegressionModel` using the training configuration specified in the previous step.

6. Serialize these `RegressionModel` objects and save them into a file.

7.3 Performance Representation

In this section, we discuss the implementation details of the `PerformanceCodec` object, which allows for representing performances using the performance codecs described in Sections 3.3.1 and 4.3.2.

The current implementation of `PerformanceCodec` allows for specifying the performance codecs described in previous chapters as follows:

- Performance Codec v. 1.0 ($\mathcal{C}_{1.0}$), which defines 4 expressive parameters (MIDI velocity, log BPR, timing and log articulation) for modeling performances in a note-wise fashion (see Section 3.3.1). This codec is best used with non-sequential models like the linear models and FFNNs.

- Performance Codec v. 1.5 ($\mathcal{C}_{1.5}$), which defines the same parameters as codec $\mathcal{C}_{1.0}$, but uses a smoothed version of the local beat period to compute parameters describing expressive tempo, timing and articulation, such that the resulting parameters are more psychologically plausible (see Appendix B).

- Performance Codec v. 2.0 ($\mathcal{C}_{2.0}$), which defines 2 onset-wise expressive parameters (MIDI velocity trend and log BPR) and 3 note-wise parameters (MIDI velocity deviations, timing and log articulation) (see Section 4.3.2). These codec is best used with sequential models for onset-wise parameters (like RNNs) and non-sequential models (linear models or FFNNs) for note-wise parameters.

- Performance Codec v. 2.5 ($\mathcal{C}_{2.5}$), which describes the same parameters as $\mathcal{C}_{2.0}$, but uses a smoothed version of local beat period used in codec $\mathcal{C}_{1.5}$ (see Appendix B).

The representation of a performance in terms of the above expressive parameters for both versions of the codec is lossless up to the average beat period (BP_{ave}), which means that an encoding of a MIDI performance in terms of these parameters allows for an exact reconstruction of the

performance, given the score and the BP_{ave}. In the current implementation, we allow the user to specify the average beats per minute, given by $\text{bpm}_{ave} = \frac{60\,\text{s}}{\text{BP}_{ave}}$, since this parameter is more intuitive than the average beat period.

An instance of the `PerformanceCodec` object has an `encode` and `decode` method, which implement the performance representation model $\mathcal{Y}$ and its corresponding performance decoder $\mathcal{Y}^{-1}$, respectively.

We describe these functions in the following subsections.

7.3.1 Encoding a Performance

Before continuing with our description of the `encode` method, we explain the way a matched performance is represented. As mentioned in Section 2.4, the datasets consisting of computer-controlled piano performances used in this thesis (the Batik/Mozart, Magaloff/Chopin and Zeilinger/Beethoven datasets) are stored in Matchfile format. This format was created by Gerhard Widmer and his colleagues for the specific purpose of matching MIDI performances to their corresponding elements in the score. Figure 7.3 shows an excerpt of the first movement of Sonata KV. 332 by W. A. Mozart performed by Roland Batik in Matchfile format. In this format, each line represents a note in the performance (see the blue rectangle in the first line in Figure 7.3) and its corresponding note in the score (see the pink rectangle in the first line in Figure 7.3). Notes added by the performer that do not appear in the score are called *inserts* (highlighted in orange in Figure 7.3). Notes in the score that were not performed are denoted as *deletions* (highlighted in green in Figure 7.3).

We parse a matched performance $\mathfrak{P}_{\mathfrak{S}}$ in Matchfile format into a `MatchFile` object, that describes the performance information. This `MatchFile` object contains `Note` and `Snote` objects, which contain information about performed notes and notes in the score respectively. Additionally, `MatchFile` objects have a `note_pairs` method, which retrieves the matching pairs of `Note` and `Snote` objects. `Note` objects have attributes indicating the performed MIDI pitch, MIDI velocity, onset and offset time in seconds. `Snote` objects have attributes indicating the MIDI pitch, enharmonic pitch and octave, score position and duration in beats. As discussed in Section 3.2, notes have a key-off and a sound-off time. In this work we refer to the offset time of a note as the latter. In the computation of the offset time of a note, we read the sustain pedal information from Matchfiles and infer the sound-off times by taking into account the sustain pedal events.

The `encode` method receives a matched performance and its corresponding score as inputs and returns a performance representation $\mathbf{T}$. For performance codec $\mathcal{C}_{1.0}$, $\mathbf{t}$ is represented as a Numpy array of shape $(N_x, 4)$, where N_x is the number of notes in the score, such that each column corresponds to an expressive parameter (in the current implementation, the order is MIDI velocity, log BPR, timing and log articulation). For Performance Codec v. 2.0, $\mathbf{T}$ consists of two arrays:

1. $\mathbf{T}_{\text{onset-wise}}$, a Numpy array of shape $(N_o, 2)$, where N_o is the number of score onsets o in the score $\mathfrak{S}$, and each column is an onset-wise parameter (in the current implementation, the order is MIDI velocity trend and log BPR); and

2. $\mathbf{T}_{\text{note-wise}}$, a Numpy array of shape $(N_x, 3)$, where the each column is a note-wise parameter (MIDI velocity deviations, timing and log articulation in the current implementation).

```
snote(n1,[f,n],3,1:1,0,1/8,0.0,0.5,[])-note(1,[f,n],3,671,1564,1564,40).
snote(n2,[f,n],4,1:1,0,2/4,0.0,2.0,[s,ls])-note(2,[f,n],4,676,1404,1480,59).
snote(n3,[a,n],3,1:1,2/16,1/8,0.5,1.0,[])-note(3,[a,n],3,892,957,1480,35).
snote(n4,[c,n],4,1:2,0,1/8,1.0,1.5,[])-note(4,[c,n],4,1071,1164,1480,33).
snote(n5,[a,n],3,1:2,2/16,1/8,1.5,2.0,[])-note(5,[a,n],3,1239,1294,1480,31).
snote(n6,[a,n],4,1:3,0,1/4,2.0,3.0,[s,le])-note(6,[a,n],4,1385,1742,1749,70).
snote(n7,[c,n],4,1:3,0,1/8,2.0,2.5,[])-note(7,[c,n],4,1455,1548,1548,16).
                                    .
                                    .
                                    .
snote(n991,[g,n],3,89:3,2/16,1/16,266.5,266.75,[])-note(991,[g,n],3,95242,95300,95300,56).
insertion-note(992,[b,n],5,95248,95275,95275,53).
snote(n993,[a,n],5,89:3,3/16,1/32,266.75,266.875,[s])-note(993,[a,n],5,95284,95336,95336,73).
snote(n993,[c,n],4,89:3,3/16,1/16,266.75,267.0,[])-deletion.
snote(n994,[g,n],5,89:3,7/32,1/32,266.875,267.0,[s])-note(994,[g,n],5,95379,95445,95445,79).
```

Figure 7.3: Excerpt of the first movement of Sonata KV. 332 by W. A. Mozart performed by Roland Batik in Matchfile format. Each line represents a score note (highlighted in pink in the first line) and a performance note (highlighted in blue for the first line). The score note information includes attributes such as the index of the (see `n2`, which corresponds to n_2 in the notation used in this thesis, highlighted in gold), its notated pitch and octave (F natural 4 , highlighted in brown and gray, respectively for n_2). The performed note information includes the number of the note in the MIDI file (2 highlighted in red for n_2), the performed onset time (in MIDI ticks, highlighted in light green for n_2); key-off time and the sound-off time estimated using pedal information (highlighted in purple and yellow for n_2, respectively); and the performed MIDI velocity of the note (highlighted in light blue). Performed notes that do not represent notes in the score are denoted as insertions (highlighted in orange). Omitted notes are denoted as deletions (see the omission of n_{933} highlighted in green).

7.3.2 Decoding a Performance

As mentioned above, in addition to defining a method for encoding performances, the `PerformanceCodec` class includes a method for decoding the prediction of a performance, $\mathbf{Y}$, generated by the predictive function of the BM (and implemented in terms of `RegressionModel` objects described below), and writing this performance into a MIDI file. In particular, the `decode` method of the `PerformanceCodec` object implements the performance decoder corresponding to the specified performance codec ($\mathcal{Y}_{1.0}^{-1}$ and $\mathcal{Y}_{2.0}^{-1}$ for performance codecs $\mathcal{C}_{1.0}$ and $\mathcal{C}_{2.0}$, respectively). This method receives the predicted performance representation $\mathbf{Y}$ and its corresponding score (represented using the methods described below), as well as the specified average bpm, to generate an expressive performance, which is written into a standard MIDI file. For Performance Codec v. 1.0, the `decode` method implements the performance decoder specified in Algorithm 3.1. For Performance Codec v. 2.0, the `decode` method implements the performance decoder specified in Algorithm 4.1. Additionally, the `decode` method includes subroutines to ensure that the generated performance is a valid MIDI file, such as ensuring that the MIDI velocities of each note are integers between 0 and 127.

7.4 Score Representation

In this section, the implementation details of a score representation model $\mathcal{F}$ are discussed. As described in Section 3.2, this representation model is defined in terms of basis functions, numerical encodings of a variety of descriptors of a musical score. As stated above, the Basis Mixer implements score representation models through `BasisGenerator` objects, and uses `Score` objects to hold score information.

7.4.1 Implementation Details

A score $\mathfrak{S}$ in MusicXML format is parsed into a `Score` object that describes a *score ontology*, i.e. a formal representation of concepts and properties relating to the information contained by a musical score. This `Score` object consists of `ScorePart` objects, which relate to the concept of staves in a musical score. The most common example would be to represent each instrument in a score by its own `ScorePart`[6]. Each of these `ScorePart` objects contains a `TimeLine` object, which acts as a "clothes line" for the elements in a musical score such as bars, notes, performance directives (e.g. slurs, dynamics and tempo markings), key and time signatures, etc. `TimeLine` objects contain a sequence of `TimePoint` objects which "pin" the score elements to a score position `t`. In this way, `TimePoint` objects contain objects (representing notes, bars, performance directives, etc.) that start and end at score position `t`.

Performance directives are parsed using a context free grammar using the Ply library[7]. In this way, performance directives represented by strings of text are tokenized, which allows us, for example, to identify both **Allegro molto** and **Allegro con brio** as instances of **Allegro**. Additionally, given that the scores from the Magaloff/Chopin and Zeilinger/Beethoven datasets were digitized using optical music recognition, we have a dictionary of equivalences from incorrectly identified textual performance directives (e.g. ***dinn.*** $\mapsto$ ***dim.***).

Each group of basis functions is implemented as a class `Basis`, which has a method `makeBasis` for computing the value of the basis functions given a `ScorePart` object. The `makeBasis` method returns two objects: an instance of `FeatureBasis`, and a list denoted as `names`. `FeatureBasis` is a class with attributes `W`, which stores the value of the basis functions evaluated for their relevant score elements as a Numpy array of shape (N, m), where N is the number of relevant elements (onsets or notes) in the `TimeLine` object and m is the number of basis functions defined by class `Basis`. On the other hand, `names` is a list stores the names of the basis functions as strings.

A user can specify a complete score representation model $\mathcal{Y}$ by specifying the basis functions (individually or by group) for each expressive parameter in a configuration file[8]. Given such a configuration, an instance of the `BasisGenerator` class generates a score representation $\mathbf{\Phi}^{(j)}$ for the j-th expressive parameter. This score representation is stored as a Numpy array of shape (N, M_j) resulting from concatenating the arrays corresponding to the basis functions defined by all `FeatureBasis` objects generated for the j-th expressive parameter, where M_j is the total number of basis functions. In case of joint modeling of expressive parameters, the basis functions for each expressive parameters are combined into a single array.

[6] Note that given the variations in the MusicXML format, in certain cases, the upper and lower staves of a piano score could be represented as individual `ScorePart` objects.
[7] http://www.dabeaz.com/ply/.
[8] A JSON file in the current implementation.

7.5 Predictive Function

In this section we describe the way that the predictive functions of the BM framework are implemented in the Basis Mixer.

7.5.1 Functions for Onset-wise Parameters

Given the sequential nature of the onset-wise expressive parameters, we use RNNs to model dynamics and tempo. Our current implementation allows us to use vanilla recurrent layers, but also more sophisticated recurrent architectures, such as long-short term memory networks (LSTMs) (Hochreiter and Schmidhuber, 1997) and gated recurrent units (GRUs) with or without multiplicative integration (Wu et al., 2016). Furthermore, these architectures can be used to build probabilistic neural networks, such as the Gaussian mixture density RNNs (GMDRNNs) described in Appendix D.

7.5.2 Functions for Note-wise Parameters

On the other hand, to model note-wise parameters, we have implemented both linear models described in Chapter 3, namely the deterministic linear model described in Section 3.3.3 and the Bayesian linear model described in Section 3.4.2. We also have implemented the non-linear models (FFNNs) described in Section 4.2.2. Furthermore, we can use FFNNs to build probabilistic neural networks such as the Gaussian mixture density FFNNs (GMDFFNNs) described in Appendix D.

7.5.3 Implementation Details

As discussed in Section 7.2, the base object implementing the predictive functions of the BM is `RegressionModel`. An instance of `RegressionModel` has attributes `input_names`, `output_names`, `fit` and `predict`. Attributes `input_names` and `output_names` are lists containing the name of the input basis functions and the name of the output parameters that the `RegressionModel` is predicting. This information can then be used to initialize a `BasisGenerator` for the purpose of rendering expressive performances, as described in Section 7.2.1. Each instance of `RegressionModel` can predict a single expressive parameter or a combination of expressive parameters, if they are modeled jointly.

Our current implementation allows for using any combination of models for predicting expressive parameters, including linear models and the recurrent networks described above.

For training linear models, we use the LSMR, an iterative algorithm for solving sparse linear least squares problems. We use the implementation of LSMR provided in `scipy.sparse.linalg`. Neural networks are constructed and trained using Lasagne (Dieleman et al., 2015), a lightweight library for building and training neural network models based on Theano (Al-Rfou et al., 2016), a Python framework for fast computation of mathematical expressions. After a model is trained, the `RegressionModel` instance containing it is serialized.

The main script for generating a rendering of a score loads the serialized instances of the `RegressionModel` object, and in case there is no model for an expressive parameter, a dummy model is used, which returns the parameters equivalent to a deadpan version.

There are several alternatives for predicting a performance:

- Return the deterministic output of the model (for deterministic models such as the FFNNs and RNNs).
- Return the expected value of the probability distribution (for probabilistic models such as the Bayesian linear model described in Section 3.4, or for GMDNFFNNs and GMDRNNs).
- Return the largest component (for GMDNFFNNs and GMDNRNNs)[9].
- Sample a performance from a generative distribution. Although this method is not yet implemented in the current version of the Basis Mixer, a description of methods for sampling performances is provided in Appendix G.

Models can only predict a score that is represented using the basis models the model was trained on. If the score does not include elements for computing a basis function, their values are set to zero.

The output of the predictive models (i.e. the predicted performance) is standardized to have zero-mean and unit variance. The user can then specify the scaling and center of the parameters via a configuration file (see the "Render Config" in Figure 7.1), or through the sliders in the web interface described in Section 7.6. This step is represented by the yellow "Post-processing" module in Figure 7.1.

7.6 Interactive Web Interface

In addition to the general implementation of the Basis Mixer discussed above, we present the prototype of a web app, which we call *The Basis Mixer: A Computational Romantic Pianist*[10]. This web app consists of an interactive user interface (UI) for the Basis Mixer that allows the user to upload a score in MusicXML format and create an expressive interpretation of the score as a MIDI and audio file. The UI is shown in Figure 7.4. For rendering the performance as an audio file we use Fluidsynth[11], an open source software synthesizer based on the SoundFont 2 specifications[12].

The Web app generates a performance using models trained using the Magaloff/Chopin and Beethoven/Zeilinger datasets, i.e. Classical and Romantic piano music from the late 18th and 19th centuries (hence the name). Using the controls in the UI, the user can specify the rendering model (see the "Rendering Model" option in the UI shown in Figure 7.4) from a list of trained models with different architectures, trained for predicting expressive parameters independently or jointly, and different training sets (e.g. using only the Magaloff/Chopin, the Zeilinger/Beethoven or both).

Additionally, there are controls in the UI (see the faders shown in Figure 7.4) for specifying the mean value and standard deviation[13] of the different expressive parameters, thereby controlling the overall expressive characteristics of the performance. The faders in the UI provide the Basis Mixer the information for post processing the predicted performance, which would be provided in a configuration file in the main implementation of the Basis Mixer (see the blue "Render

[9] In general, for Gaussian mixture models, the largest component is not guaranteed to be the largest mode of the distribution (Carreira-Perpiñán, 2000).

[10] http://lrn2cre8.ofai.at/expression-models/basis_mixer_app/static/app.html.

[11] http://www.fluidsynth.org.

[12] SoundFont refers both to a file format and its associated technology for sample-based synthesis (E-mu Systems, Inc., 2006).

[13] referred to as *degree of variation* in UI, since we believe that this term might be more intuitive for the user.

Figure 7.4: Prototype UI for the Web App.

Config" file depicted in the schematic representation in Figure 7.1). In the UI, we rename the expressive parameters to terms that we believe are more intuitive for the user:

- *Tempo* refers to log BPR;
- *Timing* refers to timing;
- *Loudness* refers to MIDI velocity trend;
- *Loudness spreading* refers to MIDI velocity deviations; and
- *Articulation* refers to log articulation.

Furthermore, there is a checkbox in the UI that allows the user to include grace notes in the score. Since grace notes do not have an explicit duration, they are "expanded", meaning that they "steal" duration from the main note that they belong to. The user can control how much duration grace notes steal from their main note using a fader.

7.7 Conclusions

This chapter describes an implementation of the BM framework, called the Basis Mixer. This implementation allows for generating expressive music performances given a score in MusicXML format. Additionally, we present the prototype of an interactive interface that allows the user to upload a score and control certain aspects of the rendering.

Future work includes using adapting the Basis Mixer to an interactive scenario, where a user

can control certain aspects of the expressive performance (such as loudness and tempo) using an interface such as a MIDI theremin, or hand controller such as the Leap Motion[14].

The implementation of the Basis Mixer described in this chapter renders an expressive performance in an offline fashion. Chapter 8 describes an accompaniment system that adapts the Basis Mixer to allow for expressive accompaniment in real-time.

[14] https://www.leapmotion.com.

8 Using Basis Function Models for Expressive Accompaniment: The ACCompanion

This chapter contains material published in

- Cancino-Chacón, C., Bonev, M., Durand, A., Grachten, M., Arzt, A., Bishop, L., Goebl, W., and Widmer, G. (2017a). The ACCompanion v0.1: An Expressive Accompaniment System. In *Late Breaking/ Demo, 18th International Society for Music Information Retrieval Conference (ISMIR 2017)*, Suzhou, China

8.1 Introduction

In this chapter, we present a preliminary version of an expressive automatic accompaniment system capable of adapting aspects of the accompaniment performance to the current playing style of the soloist.

So far, the BM framework described in this thesis has focused on analysis and generation of solo performances. In Chapter 5 we describe an extension of the BM for modeling ensemble performances, but this approach is mostly aimed at analyzing and comparing (recorded) performances of a piece, not for the purpose of generating a new rendering of the piece. Nevertheless, in recent years there has been a growing interest in developing human–computer interaction systems that generate expressive music performances. As generative models, these interactive systems allow humans to influence a performance either by generating their own expression (either accompanying or improvising with the user), or by allowing the user to shape certain aspects of the performance in real time (i.e. allowing the user to *conduct* the performance). Furthermore, as analysis tools, these models can be used to gain knowledge about the way humans interact with each other.

Rowe (1992) proposed a terminology for categorizing interactive music systems in three dimensions:

1. *score-driven* vs. *performance-driven*, referring to whether the system follows a musical score or responds to a human performance;
2. *instrument paradigm* vs. *player paradigm*, if the system is meant for solo or ensemble performances;
3. *transformative* vs. *generative* vs. *sequenced*, describing how the system renders the music.

For more thorough review of the current state-of-the-art in expressive interactive systems see Section 9.3.2 and (Chew and McPherson, 2017).

According to Rowe's taxonomy, accompaniment systems are score-driven, player paradigm systems that automatically generate synchronized accompaniment for a (human) solo performance, (usually) in real time.

Dannenberg (1984) identifies three tasks that accompaniment systems must solve in order to successfully perform together with a human:

1. *Detecting the solo part*, i.e. capturing a human performance in real time (either from a microphone or a MIDI instrument) and identifying the performed notes.
2. *Matching the detected input to the score*, i.e. matching these performed notes to notes in the score (also in the presence of errors). The first two tasks are commonly referred to as real-time *score following*, and a system for solving this task is referred to as a *score follower* (SF).
3. *Generating an expressive accompaniment part.*

While most of the work on accompaniment systems has focused on the problem of score following (Cont et al., 2012; Nakamura et al., 2015a; Raphael, 2010), in recent years there has been increased interest in exploring expressive accompaniment systems. For example, Xia et al. (2015) present an accompaniment system capable of generating expressive dynamics and timing using linear dynamical systems.

In this chapter, we present a preliminary version of an expressive accompaniment system for MIDI input, which we call *the ACCompanion*. This system combines a monophonic SF based on Hidden Markov Models (HMMs) with a version of the Basis Mixer described in Chapter 7 adapted for the accompaniment scenario. In its current state, the system can react to the playing style of the soloist, in particular in terms of the overall dynamics, tempo and articulation. The current implementation of this system consists of three main modules that run in parallel with each other:

1. a module that implements the SF, a module that adapts the expression of the accompaniment part according to the performance style of the soloist (which we refer to as the *accompanist*); and
2. a module that handles the MIDI inputs and feeds them to the SF, and generates the MIDI messages corresponding to the accompaniment part and sends them to the output MIDI device. We refer to this module as the *MIDI handler*.

Each of these modules roughly corresponds to one of the tasks proposed by Dannenberg (1984): the MIDI producer covers the first task (and some aspects of the third task), and the SF and accompanist modules solve the second and third task, respectively. In the rest of this chapter we will focus on the SF and accompanist modules, since they are the most interesting components from the point of view of computational models of expressive performance, although we will provide some brief overview of the MIDI handler.

As mentioned in Chapter 1, the work on the ACCompanion is part of an inter-institutional collaboration involving the Austrian Research Institute for Artificial Intelligence, the Institute of Computational Perception at the Johannes Kepler University Linz and the Department of Musical Acoustics at the University of Music and Performing Arts Vienna. The overall design of the system was done by myself, but specific components of the system were designed by Amaury Durand and Martin Bonev. In particular, Amaury designed the HMM-based SF(described below in Section 8.2) and Martin designed a graphical interface for visualizing the performance of the solo and accompaniment parts (described below in Section 8.4). I adapted the Basis Mixer for a real-time accompaniment scenario (with occasional help from Maarten Grachten). The bulk the implementation of the system was done by Amaury, Martin Bonev and myself. We thank Jan Hajič Jr. and Anna Aljanaki for testing an early version of the prototype.

The rest of this chapter is structured as follows. Section 8.2 discusses the HMM-based monophonic SF. Section 8.3 briefly describes how to generate the expressiveness in the accompaniment.

Section 8.4 provides some technical details on the implementation of the prototype version of the ACCompanion. Finally, conclusions are provided in Section 8.5.

8.2 Score Following

This section discusses theoretical aspects of score following relevant for expressive accompaniment systems, as well as presenting a brief overview of the HMM-based monophonic SF used in the ACCompanion. As mentioned in the introduction, in our prototype this SF was designed and implemented by Amaury Durand. A detailed description of this SF is provided in (Durand, 2017).

As mentioned above, the task of score following involves both detecting the solo part and matching this part to the corresponding parts in the score. Since the ACCompanion is a system for MIDI input, the first task is easily solved, since for each performed note there is a corresponding MIDI note, which as described in Section 3.2, can be described with four parameters: its MIDI pitch, MIDI velocity, onset time and duration.

The rest of this section is structured as follows: Section 8.2.1 provides a brief overview of the score following task, focusing on aspects relevant to expressive accompaniment. Section 8.2.2 describes the HMM-based score follower used in the ACCompanion.

8.2.1 Description of the Score Following Problem for Expressive Accompaniment

In the typical scenario of a duet performance, we can consider that a duet score $\mathfrak{S}$ consists of two parts, $\mathfrak{S}^{(s)}$ and $\mathfrak{S}^{(a)}$ which correspond to the solo and accompaniment parts, respectively. The sets of notes in the solo and accompaniment scores are denoted as $\boldsymbol{\mathcal{X}}^{(s)} = \{n_i^{(s)} \mid n_i^{(s)} \in \mathfrak{S}^{(s)}\}$ and $\boldsymbol{\mathcal{X}}^{(a)} = \{n_i^{(a)} \mid n_i^{(a)} \in \mathfrak{S}^{(a)}\}$, respectively. In a similar way, the sets of onsets in the solo and accompaniment scores are denoted as $\boldsymbol{\mathcal{O}}^{(s)} = \{o_i^{(s)} \mid o_i^{(s)} \in \mathfrak{S}^{(s)}\}$ and $\boldsymbol{\mathcal{O}}^{(a)} = \{o_i^{(a)} \mid o_i^{(a)} \in \mathfrak{S}^{(a)}\}$, respectively. The equivalent performed onset[1] corresponding to the i-th score position in the accompaniment and solo scores will be denoted as $\hat{o}_{i;a}^{perf}$ and $\hat{o}_{i;s}^{perf}$, respectively. In this chapter, we use a slightly different formulation of the inter-onset interval (IOI) than the one used throughout this thesis. Instead of associating the IOI to the *beginning of the interval* (i.e. $\text{IOI}(o_i) = \text{onset}(o_{i+1}) - \text{onset}(o_i)$, as discussed in Section 3.3.1), we associate the IOI to the *end of the interval*, i.e. $\text{IOI}(o_i) = \text{onset}(o_i) - \text{onset}(o_{i-1})$. This change is required in the case of real-time causal accompaniment systems, since at a given point in time we only know when the last onset was performed and not when the next onset will be performed.

Furthermore, since we are assuming that the solo part is performed in a MIDI instrument, we can represent the performance of the solo part with $\boldsymbol{\mathcal{X}}_{perf}^{(s)} = \{n_i^{(p)} \mid n_i^{(p)} \in \mathfrak{P}^{(s)}\}$, where $n_i^{(p)}$ is the i-th performed note, and $|\boldsymbol{\mathcal{X}}_{perf}^{(s)}| = N_p$ is the total number of performed notes. In this case, the onset and duration of performed solo note $n^{(p)}$ are given in seconds and not in beats as is the case of score notes. Since we are assuming that the solo part is monophonic, the number of notes in the solo score is equal to the number of score onsets in the solo score, i.e. $N_x^{(s)} = N_o^{(s)}$. We will slightly abuse notation and denote $\hat{o}_{i;p}^{perf}$ as the onset time (in seconds) of the notes performed by the soloist[2].

[1] an aggregate of the performed onset times of all notes belonging to that score onset. See Section 3.3.1.

[2] Note that even if the solo part is monophonic, the i-th performed onset $\hat{o}_{i;p}^{perf}$ might not correspond to the i-th equivalent performed onset of the solo part $\hat{o}_{i;s}^{perf}$, since the latter represents the performance information that has been matched to an element of the score of the solo part, whereas the first one refers to the notes that the

The set of notes performed by the soloist $\boldsymbol{\mathcal{X}}^{(s)}_{perf}$ might contain notes that do not correspond to any note in $\boldsymbol{\mathcal{X}}^{(s)}$, which we will refer to as *insertions*. Conversely, a note in $\boldsymbol{\mathcal{X}}^{(s)}$ omitted during $\mathfrak{P}^{(s)}$ will be referred to as a *skip*. With $n^{(p)}_{i^*}$, we denote the note in the set of performed notes $\boldsymbol{\mathcal{X}}^{(s)}_{perf}$ that corresponds to score note $n^{(s)}_i \in \boldsymbol{\mathcal{X}}^{(s)}$; and with $n^{(s)}_{i^*}$, we denote the note in $\boldsymbol{\mathcal{X}}^{(s)}$ that corresponds to note $n^{(p)}_i \in \boldsymbol{\mathcal{X}}^{(s)}_{perf}$. It is important to note the real-time aspect of the score-following process. In contrast to the offline performance rendering considered in previous chapters, there is no complete match of the performance of the solo part to its score, but a partial one that is updated every time that there is a new note performed by the soloist. Using the idea of matched performance from Definition 8.1, we can define an SF as follows:

Definition 8.1 (Score Follower)**.** A *score follower* (SF) is a function sfollower: $\mathfrak{P}^{(s)} \mapsto \mathfrak{S}^{(s)}$ that maps events in the performance of the solo part (in this case performed MIDI notes) to their corresponding elements in the score in real-time. We denote this function as

$$\text{sfollower}\left(\mathfrak{P}^{(s)}, \tau \mid \mathfrak{S}^{(s)}\right) = \mathfrak{P}^{(s;\tau)}_{\mathfrak{S}}, \tag{8.1}$$

where $\mathfrak{P}^{(s;\tau)}_{\mathfrak{S}} = \left\{\left(n^{(s)}_i, n^{(p)}_{i^*}\right) \mid n^{(s)}_i \in \boldsymbol{\mathcal{X}}^{(s)} \text{ and } n^{(p)}_{i^*} \in \boldsymbol{\mathcal{X}}^{(s)}_{perf} \text{ such that } \text{onset}_{perf}\left(n^{(p)}_{i^*}\right) \leq \tau \ \forall i\right\}$ is a partially matched performance up to (absolute) performance time τ in seconds.

The above definition describes an SF that matches notes in the performance to notes in the score, i.e. a partial *note-wise* performance-to-score matcher, instead of simply matching events of the performance to score positions (i.e. in an *onset-wise* fashion). Having such a note-wise SF is a desirable feature for the case of an expressive accompaniment system, since it allows the accompaniment system to adapt its performance of the accompaniment part to aspects of the performance style of the soloist that are inherently defined at the note level (such as articulation, which requires the performed duration of the individual notes).

Since we are only considering the case of monophonic SFs, we will abuse notation and write the matching of a performed note $n^{(p)}_i$ to a note in the solo part as

$$\text{sfollower}\left(n^{(p)}_i\right) = \begin{cases} n^{(s)}_{i^*} & \text{if the SF believes } n^{(p)}_i \text{ to correspond to score note } n^{(s)}_{i^*} \\ \text{insertion} & \text{otherwise.} \end{cases} \tag{8.2}$$

In the above equation we say that the SF "believes" that a performed note corresponds to a score note, instead of saying that the performed note *actually corresponds* to a score note, to emphasize that the sfollower function produces an estimate of the note in the score to which a performed note should be matched (i.e. the score follower can incorrectly match a performed note). In the case of a real-time accompaniment, SFs with high accuracies are required[3].

8.2.2 HMM-based Score Follower

In this section we provide a brief overview of the probabilistic SF used in the ACCompanion.

The SF used in the ACCompanion is based on the switching Kalman filter, a hybrid probabilistic model which combines an HMM and a Kalman filter (i.e. a linear dynamical system), whose

soloist performs, including ornaments (which are not considered in this version of the system) and mistakes.

[3]How accurate an SF must be for the purpose of expressive accompaniment is an open research question.

parameters depend on the states of the HMM (Murphy, 1998). In the rest of this discussion, we refer to this model as *Kalman HMM* (KHMM).

An HMM is a state space model defined by two discrete processes: a latent (hidden) process, which is assumed to be a Markov chain, and an observed process, which depends on the latent process (Rabiner, 1989). A Kalman filter can be thought of as a special kind of HMM where both latent and observed processes are continuous and have linear dependencies (Bishop, 2006).

Using the terminology from HMMs, in the setup of an accompaniment system we *observe* a sequence of notes $n_0^{(p)}, n_i^{(p)}, \ldots, n_{N_p}^{(p)}$, each of which corresponds to a position in the solo score, which we want to estimate (since it is *hidden*).

In the following, we describe the state space of the KHMM:

Observed Variables:

- P_i is a discrete random variable representing the MIDI pitch of performed note $n_i^{(p)}$.
- IOI_i is a continuous random variable representing the IOI between $n_i^{(p)}$ and $n_{i-1}^{(p)}$.

Latent Variables:

- O_i is a discrete random variable representing the index of the onset in the solo score corresponding to the observed performed note $n_i^{(p)}$[4]. We let O_i take integer values between in 0 and $N_o^{(s)} - 1$, and all half-integers in between (e.g. 0.5, 1.5, 2.5). This set of values is denoted as $\mathcal{Q} = \left\{0, 0.5, \ldots, N_o^{(s)} - 1\right\}$. Integer values correspond to the indices of onsets in the solo score (e.g. $O_i = j$ means that the KHMM assigns $n_i^{(p)}$ to score onset $o_j^{(s)}$). Half-integer values correspond to insertions.
- BP_i is a continuous random variable representing the local beat period (BP) in seconds estimated for performed note $n_i^{(p)}$.

The initial states of these latent variables are denoted as O_{-1} and BP_{-1}, respectively. In the following discussion, the shorthand notation $A_{a:b}$ refers to the sequence $A_a, A_{a+1}, \ldots, A_{b-1}, A_b$.

The KHMM is completely specified by three components:

1. A *prior initial distribution* of the latent variables, denoted as $p(O_{-1})$ and $p(BP_{-1})$, respectively.
2. A *transition model* specifying the probability distribution of latent variables O_i and BP_i, respectively, given the sequence of observed and latent variables. The transition model of the KHMM can be written as

$$p(O_i \mid O_{-1:i-1}, BP_{-1:i}, IOI_{0:i-1}, P_{0:i}, IOI_{0:i}) = p(O_i \mid O_{i-1}) \tag{8.3}$$
$$p(BP_i \mid BP_{-1:i-1}, O_{-1:i}, IOI_{0:i}, P_{0:i}) = p(BP_i \mid BP_{i-1}, O_i), \tag{8.4}$$

   where $p(O_i \mid O_{i-1})$ is the transition model from the HMM and $p(BP_i \mid BP_{i-1}, O_i)$ is the transition model from the Kalman filter, whose parameters depend on the state of O_i.
3. An *observation model*, which specifies the probability distribution of the current observation given the sequence of observations and latent states. The observation model of the

[4]Since we are assuming that the solo score is strictly monophonic, each score onset in the solo part corresponds to a single note, and therefore, we can denote both score onsets and notes with the same index.

KHMM is defined as

$$p(P_i \mid P_{0:i-1}, O_{-1:i}, BP_{-1:i}, IOI_{0:i-1}) = p(P_i \mid O_i) \tag{8.5}$$
$$p(IOI_i \mid IOI_{0:i-1}, O_{-1:i}, BP_{-1:i}, P_{0:i}) = p(IOI_i \mid O_i, BP_i), \tag{8.6}$$

where $p(P_i \mid O_i)$ is the HMM part of the observation model, and uses the tempo from the latent state of the Kalman filter.

Using these components, we can write the joint probability distribution defined by the KHMM as

$$\begin{aligned} p(O_{0:N_p}, BP_{0:N_p}, P_{0:N_p}, IOI_{0:N_p}) =& p(O_{-1})p(BP_{-1}) \prod_{i=0}^{N_p-1} p(O_i \mid O_{i-1})p(BP_i \mid BP_{i-1}, O_i) \\ & \times \prod_{i=0}^{N_p-1} p(P_i \mid O_i)p(IOI_i \mid O_i, BP_i). \end{aligned} \tag{8.7}$$

The KHMM model is illustrated in Figure 8.1, where the nodes represent random variables (in the figure, square nodes represent discrete variables and circles represent continuous variables) and directed edges (arrows in the figure) indicate dependencies between the variables.

At a given position, we can use the KHMM to infer the latent states (the index of the score position and the BP of the solo) in a causal way as

$$\hat{q}_i = \underset{q_i \in \mathcal{Q}}{\operatorname{argmax}}\, p(O_i = q_i \mid P_{0:i}, IOI_{0:i}) \tag{8.8}$$
$$\hat{b}_i = \mathbb{E}\left\{BP_i \mid P_{0:i}, IOI_{0:i}, \hat{q}_i\right\} \tag{8.9}$$

where $\hat{q}_i$ is the estimated index of the score onset and estimated BP corresponding to performed note $n_i^{(p)}$, respectively. This inference of the latent states can be efficiently done using the *forward algorithm* (Rabiner, 1989; Murphy, 1998). For a more detailed description of the form of the components of the KHMM (initial prior distribution, transition and observation models) see Section 3.3 in (Durand, 2017). An explicit algorithm for inferring the latent states of the KHMM is presented in Algorithm 2.3.1 in (Durand, 2017).

Using the above estimation of the latent state representing the score onsets, we can write the SF function defined in Equation (8.2) as

$$\text{sfollower}\left(n_i^{(p)}\right) = \begin{cases} n_{\hat{q}_i}^{(s)} & \text{if } \hat{q}_i \in \mathbb{Z} \\ \text{insertion} & \text{otherwise.} \end{cases} \tag{8.10}$$

8.3 Adapting the Basis Function Models for Expressive Accompaniment

This section focuses on describing the accompanist module of the ACCompanion. In a nutshell, the accompanist module adapts the BM framework, in particular the RNBM introduced in Section 4.3, for the case of modeling expressive accompaniment.

We use RNBM to allow the ACCompanion to adjust its performance to the soloist in three steps:

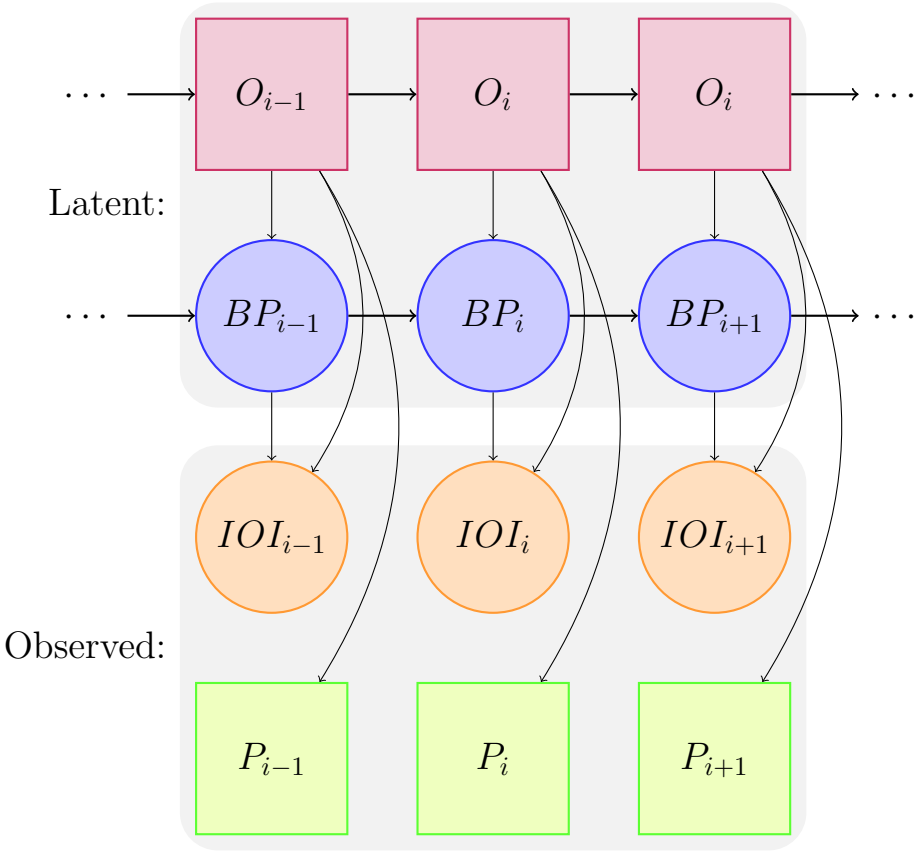


Figure 8.1: Schematic representation of the KHMM score follower. O_i (in pink) is a random variable representing the onset at observation i. BP_i (in blue) is a random variable representing the performed BP at observation i. IOI_i (in orange) is a random variable representing the observed IOI at observation i. P_i (in light green) is a random variable representing the observed MIDI pitch at observation i. Square nodes represent discrete variables and circle nodes represent continuous variables. The gray boxes denote the latent and observed variables. See text for explanation.

1. Before the performance starts, we precompute the performance of the accompaniment using the RNBM (using the Basis Mixer described in Chapter 7), and scale and/or center the predictions.
2. Every time that the SF identifies a note performed by the soloist as a note in the score, we update our estimates of the performed local dynamics, tempo and articulation (we refer to these parameters as the *local playing style* of the soloist).
3. We use the estimates of the local playing style of the soloist to rescale and/or recenter the predictions of the RNBM.

In the rest of this section, we describe each of these steps in more detail.

8.3.1 Precomputing the Performance of the Accompaniment Score

In this section we describe how to use RNBM for precomputing the performance of the accompaniment score, which will be later adapted in real-time to the performance of the soloist.

We use performance codec $\mathcal{C}_{2.5}$, defined in Appendix B for modeling an expressive performance

with RNBMs. This codec is a variant of performance codec $\mathcal{C}_{2.0}$ (described in Section 4.3.2) which uses a smoothed version of the local beat period to compute parameters describing expressive tempo, timing and articulation, such that the resulting parameters are more psychologically plausible Performance codec $\mathcal{C}_{2.5}$ describes an expressive performance using five parameters, divided into two groups:

1. two onset-wise parameters describing loudness (MIDI velocity trend) and tempo (the logarithm of the beat period ration, or log BPR); and

2. three note-wise parameters, where two parameters of these parameters describe deviations from the local dynamics and tempo for the individual notes (MIDI velocity deviations and timing, respectively); and a parameter describing the articulation of each notes (log articulation).

The predictive model of the RNBM consists of two RNNs for predicting MIDI velocity trend and log BPR, respectively, and three FFNNs for predicting MIDI velocity deviations, timing and log articulation, respectively.

The architecture of the RNNs consists of a single LSTM layer with multiplicative integration with 20 units, and a linear output. The architecture of the FFNNs consists of a single hidden layer with 20 softplus units and an output layer with a single linear unit. These architectures were chosen for their simplicity. In future work, it would be interesting to explore which neural architectures lead to better predictions. These networks were trained to minimize the squared error on the Zeilinger/Beethoven dataset using RMSProp (Tieleman and Hinton, 2012), a variant of the stochastic gradient descent algorithm (SGD) as described in Section 4.2.2. We use 80% of the pieces for training and 20% for validation. All networks were trained for a maximum of 5000 epochs with a learning rate of 10^{-3} and a gradient moving average decay of 0.9. To avoid overfitting, we use l_2-norm weight regularization with a coefficient of 10^{-3}. For FFNNs, dropout was used after the hidden layer, with a probability of 0.5. For RNNs, the value of the gradients was clipped to lie between -2 and 2. The gradients for back-propagation were truncated after 50 steps. Early stopping was used after 100 epochs without improvement in the validation set.

Using these trained networks, we generate predictions of each of the parameters as

$$f^{(\text{vel}_{trend})}(\mathbf{\Phi}^{(\text{vel}_{trend})}) = \mathbf{y}^{(\text{vel}_{trend})} \in \mathbb{R}^{N_o^{(a)}} \tag{8.11}$$

$$f^{(\log \text{bpr})}(\mathbf{\Phi}^{(\log \text{bpr})}) = \mathbf{y}^{(\log \text{bpr})} \in \mathbb{R}^{N_o^{(a)}} \tag{8.12}$$

$$f^{(\text{vel}_{dev})}(\mathbf{\Phi}^{(\text{vel}_{dev})}) = \mathbf{y}^{(\text{vel}_{dev})} \in \mathbb{R}^{N_x^{(a)}} \tag{8.13}$$

$$f^{(\text{tim})}(\mathbf{\Phi}^{(\text{tim})}) = \mathbf{y}^{(\text{tim})} \in \mathbb{R}^{N_x^{(a)}} \tag{8.14}$$

$$f^{(\log \text{art})}(\mathbf{\Phi}^{(\log \text{art})}) = \mathbf{y}^{(\log \text{art})} \in \mathbb{R}^{N_x^{(a)}} \tag{8.15}$$

where each $f^{(j)}$ and $\mathbf{\Phi}^{(j)}$ are the predictive function and the matrix representation of the accompaniment score through basis functions corresponding to the j-th expressive parameter (MIDI velocity trend, log BPR, MIDI velocity deviations, timing and log articulation), respectively. In contrast to the case of the solo performance, the accompaniment score is not assumed to be monophonic, and so the number of notes in the score is generally larger than the number of onsets in the score, i.e. $N_x^{(a)} \leq N_o^{(a)}$.

The predictions of the network are then scaled and recentered as follows:

$$\hat{\mathbf{y}}^{(\text{vel}_{trend})} = \frac{1}{\text{average}\left(\mathbf{y}^{(\text{vel}_{trend})}\right)}\mathbf{y}^{(\text{vel}_{trend})} \tag{8.16}$$

$$\hat{\mathbf{y}}^{(\log \text{bpr})} = \mathbf{y}^{(\log \text{bpr})} - \text{average}\left(\mathbf{y}^{(\log \text{bpr})}\right) \cdot \mathbf{1} \tag{8.17}$$

$$\hat{\mathbf{y}}^{(\text{vel}_{dev})} = 127 \cdot \mathbf{y}^{(\text{vel}_{dev})} \tag{8.18}$$

$$\hat{\mathbf{y}}^{(\text{tim})} = \mathbf{y}^{(\text{tim})} - \text{average}\left(\mathbf{y}^{(\text{tim})}\right) \cdot \mathbf{1} \tag{8.19}$$

$$\hat{\mathbf{y}}^{(\log \text{art})} = \mathbf{y}^{(\log \text{art})} - \text{average}\left(\mathbf{y}^{(\log \text{art})}\right) \cdot \mathbf{1} \tag{8.20}$$

where average$(\cdot)$ computes the average value of the components of its input vector, and $\mathbf{1}$ is a vector of ones of the appropriate size (i.e. the vector is in $\mathbb{R}^{N_o^{(a)}}$ for vel_{trend} and $\log \text{bpr}$ and in $\mathbb{R}^{N_x^{(a)}}$ otherwise). Although the scaling and recentering procedure described above does not constitute a proper normalization in the strict mathematical sense, for the sake of convenience we will refer to the above defined vectors as *normalized predictions of the RNBM*.

8.3.2 Capturing the Local Performance Style of the Soloist

In this section we describe how to capture the local performance style of the soloist in terms of tempo, dynamics and articulation. In the following discussion, to avoid cluttering notation, we will slightly abuse notation and refer to $o_i^{(s)}$ as both the current (matched) performed note and onset, since in the case of a monophonic solo part, each score onset consists of a single note.

Tempo

While in our preliminary tests we found that the KHMM described in Section 8.2.2 provides accurate and robust estimates of the score position of the solo part, we found that the estimates of the local tempo (see Equation (8.9)) were not very stable.

In order to address this issue we propose to use two alternative methods:

- The first of these methods is to use the instantaneous BP, computed only from the local IOI, i.e.,

$$\text{BP}_{i;\text{lioi}} = \frac{\hat{o}^{perf}\, i, s - \hat{o}^{perf}_{i-1,s}}{\text{IOI}(o_i)}, \tag{8.21}$$

  where $\text{IOI}\left(o_i^{(s)}\right) = \text{onset}\left(o_{i+1}^{(s)}\right) - \text{onset}\left(o_i^{(s)}\right)$ is the score IOI, as discussed in Section 8.2.1. We will refer to this method as *local IOI estimation*. Not surprisingly, we found in our tests that using this method leads to an extremely reactive system: the system always seems to be catching up with what the soloist is performing, which does not lead to a very rewarding playing experience. This is particularly evident (and annoying) in places where the soloist is slowing down.

- The second method involves using a linear sensorimotor synchronization (LSMS) model (Vorberg and Schulze, 2002). This model predicts the next performed onset based on the previous performed onset, an estimate of the tempo and the asynchrony between the predicted

and the onset and the actual observed onset . The updates of this model can be computed as:

$$\hat{o}^{perf}_{i,s^*} = \hat{o}^{perf}_{i-1,s^*} + \mathrm{BP}_{i-1;\mathrm{lsms}}\mathrm{IOI}\left(o_i^{(s)}\right) - \eta_a A_{i-1} \tag{8.22}$$

$$\mathrm{BP}_{i;\mathrm{lsms}} = \mathrm{BP}_{i-1;\mathrm{lsms}} + \eta_\tau A_{i-1} \tag{8.23}$$

$$A_i = \hat{o}^{perf}_{i,s^*} - \hat{o}^{perf}_{i,s}, \tag{8.24}$$

where $\hat{o}^{perf}_{i,s}$ denotes the actually observed onset and $\hat{o}^{perf}_{i,s^*}$ the onset predicted by the LSMS model. $\mathrm{BP}_{i;\mathrm{lsms}}$ denotes the estimated BP, and A_i is the asynchrony. In the above estimation, η_a and η_τ are parameters determining how much the asynchrony affects the updates of onset and tempo, respectively (small η_a and η_τ imply that the model updates the tempo very slowly, and large η_a and η_τ produce fast changes in tempo).

An issue with the LSMS method is that if the difference between the initial BPR specified to the system and the initial tempo by the soloist is large, the system takes longer to adapt, which leads to the soloist feeling that the system is not very responsive. In practice, we found out that a better strategy is to use a hybrid scheme, where the initial estimation of the tempo is done with the local IOI model (for the first onsets), and then switches to the LSMS model, i.e.,

$$\mathrm{BP}_i = \begin{cases} \mathrm{BP}_{init} & \text{if } i = 0 \\ \mathrm{BP}_{i;\mathrm{lioi}} & \text{if } 1 \leq i \leq 3 \\ \mathrm{BP}_{i;\mathrm{lsms}} & \text{otherwise.} \end{cases} \tag{8.25}$$

where BP_{init} is the initial tempo.

Determining which tempo model is best for the case of expressive accompaniment systems is an open research question. Most research in accompaniment systems tend to use (Gaussian) linear models (Raphael, 2001a, 2003; Cont et al., 2012; Nakamura et al., 2013; Xia and Dannenberg, 2015; Burloiu, 2016; Maezawa and Yamamoto, 2016; Cancino-Chacón et al., 2017a). Future work might involve experimenting with more sophisticated sensorimotor models of synchronization.

Dynamics

For capturing the dynamics performed by the soloist, we use a running average estimation of the MIDI velocity of the solo part. This average MIDI velocity is computed as

$$\mathrm{csvel}_i = \mathrm{csvel}_{i-1} + \mathrm{vel}\left(o_i^{(s)}\right) \tag{8.26}$$

$$\mathrm{vel}_i = \frac{\mathrm{csvel}_i - \mathrm{csvel}_{i-k_{\mathrm{vel}}}}{k_{\mathrm{vel}}} \tag{8.27}$$

where csvel_i is a term that iteratively computes the cumulative sum of the performed MIDI velocities (we initialize to $\mathrm{csvel}_{-1} = 0$), and $\mathrm{vel}\left(o_i^{(s)}\right)$ is the MIDI velocity of the latest performed note that was matched to an onset in the solo score. The parameter $k_{\mathrm{vel}} \geq 1 \in \mathbb{Z}$ is an integer controlling the smoothness of the curve of MIDI velocity. A large k_{vel} produces a very smooth MIDI velocity curve, and $k_{\mathrm{vel}} = 1$ produces the performed MIDI velocity of the current note in the solo part. In our experiments we set $k_{\mathrm{vel}} = 3$.

This method of computing average MIDI velocity means that if the soloist were to make a single (loud) accent, the accompaniment would not overcompensate by playing too loudly.

Articulation

The articulation is estimated using a running average of the articulation ratio, computed as

$$\text{csart}_i = \text{csart}_{i-1} + \frac{\text{duration}_{perf}\left(o_i^{(s)}\right)}{\text{BP}_i \times \text{duration}\left(o_i^{(s)}\right)} \tag{8.28}$$

$$\text{art}_i = \frac{\text{csart}_i - \text{art}_{k_{\text{art}}}}{k_{\text{art}}} \tag{8.29}$$

where $\text{duration}_{perf}\left(o_i^{(s)}\right)$ and $\text{duration}\left(o_i^{(s)}\right)$ are the performed and notated duration of the latest performed note[5]; BP_i is the current estimate of the local tempo (computed using Equation 8.25), and csart_i is a term that iteratively computes the cumulative sum of the articulation ratio (and is initialized to $\text{csart}_{-1} = 0$). As described for k_{vel}, the parameter $k_{\text{art}} \geq 1 \in \mathbb{Z}$ is an integer controlling the smoothness of the articulation curve. In our experiments we set $k_{\text{art}} = 3$.

Similar to the case of the average MIDI velocity described above, this method for computing the articulation ratio means that if the soloist were to make a single note staccato in an otherwise legato section, the accompaniment would not also suddenly be performed staccato.

8.3.3 Adapting the Performance of the Accompaniment Part to the Soloist

Finally, in this section we describe how to combine the precomputed normalized prediction of the performance of the accompaniment score described in Section 8.3.1 and the local performance style of the soloist described in Section 8.3.2 to adapt the performance of the accompaniment to the soloist.

In the following discussion, $\hat{\mathbf{y}}\left[o_k^{(a)}\right]$ will denote the components of a vector of normalized predictions corresponding to the notes in score onset $o_k^{(a)}$ in the accompaniment part. The i-th component of $\hat{\mathbf{y}}\left[o_k^{(a)}\right]$ will be denoted as $\hat{y}_l\left[o_k^{(a)}\right]$, which corresponds to the l-th note belonging to $o_k^{(a)}$, which we denote as $n_{k_l}^{(a)}$. For the sake of keeping notation consistent, $\hat{\mathbf{y}}\left[o_k^{(a)}\right]$ will always be written as boldface, even if this quantity is a scalar, as is the case for the onset-wise parameters or for score onsets corresponding to a single note in the case of note-wise parameters. Furthermore, in order to avoid confusions with different indices denoting different score positions, in this section we will use index i to exclusively denote the index of the latest performed note in the solo part. Given the latest matched performed note of the solo part $o_i^{(s)}$, performed at onset time $\hat{o}_{i,s}^{perf}$, we first need to identify the *set of upcoming onsets* in the accompaniment score, which we denote by

$$\boldsymbol{\mathcal{O}}_{next}^{(a)}\left(o_i^{(s)}\right) = \left\{o_k^{(a)} \mid \text{onset}\left(o_k^{(a)}\right) \geq \text{onset}\left(o_i^{(s)}\right)\right\}. \tag{8.30}$$

In the rest of this section, we describe how to adapt the normalized predictions of the RNBM to the local playing style of the soloist.

Tempo, Timing and Articulation

In order to generate the onsets times of the notes in the accompaniment, we use the estimated tempo from the soloist to scale the tempo predictions of the RNBM and generate equivalent

[5] See discussion about the importance of note-wise matching for score following in Section 8.2.1.

onset times for each onset. Afterwards we use the timing and articulation predictions of the RNBM to predict the equivalent onset times and durations for each note in the accompaniment score onsets, respectively. This procedure is described as follows:

By definition, the first onset in the set of upcoming onsets has either the same score position as $o_i^{(s)}$, or occurs afterwards. The equivalent onset time for this onset, which we denote as $o_{i_0}^{(a)}$ is given as

$$\hat{o}_{i_0;a}^{perf} = \hat{o}_{i;s}^{perf} + 2^{\hat{\mathbf{y}}^{(\log \mathrm{bpr})}\left[o_{i_0}^{(a)}\right]} \cdot \mathrm{BP}_i \cdot \left(\mathrm{onset}\left(o_{i_0}^{(a)}\right) - \mathrm{onset}\left(o_i^{(0)}\right)\right), \tag{8.31}$$

where BP_i is the estimated tempo of the soloist, computed as described in Section 8.3.2 using Equation 8.25. The above equation implies that if both $o_{i_0}^{(a)}$ and $o_i^{(s)}$ have the same score position, they have exactly the same performance onset time. Note that this property works only if the accompaniment is being reproduced in a MIDI device, where the time between receiving the MIDI message of the solo performance and sending the message for the accompaniment note is negligible. The equivalent onset times for the rest of the onsets in $\boldsymbol{\mathcal{O}}_{next}^{(a)}\left(o_i^{(s)}\right)$ (i.e. onsets with indices $j > i_0$) is given by

$$\hat{o}_{j;a}^{perf} = \hat{o}_{j-1;a}^{perf} + 2^{\hat{\mathbf{y}}^{(\log \mathrm{bpr})}\left[o_j^{(a)}\right]} \cdot \mathrm{BP}_i \cdot \left(\mathrm{onset}\left(o_j^{(a)}\right) - \mathrm{onset}\left(o_{j-1}^{(a)}\right)\right). \tag{8.32}$$

Using the timing predictions of the RNBM, the onset time of the k-th note of onset $o_j^{(a)}$ (denoted as $\hat{o}_{j_k;a}^{perf}$) in the set of upcoming onsets can be estimated as

$$\hat{o}_{j_k;a}^{perf} = \max\left(\hat{o}_{i;s}^{perf}\,;\, \hat{o}_{j_k;a}^{perf} - \hat{y}_k^{(\mathrm{tim})}\left[o_j^{(a)}\right]\right) \tag{8.33}$$

where $\max(\cdot\,;\cdot)$ returns the maximum argument. The inclusion of this operator is motivated by the fact that the timing deviations allow a note to be played before or after the equivalent onset. Therefore it could happen that the onset time of a note is estimated to occur at a time before the current observed onset (i.e. $\hat{o}_{i;s}^{perf}$). The current version of the ACCompanion cannot time-travel to the past to perform a note before the current (absolute) onset time[6].

In a similar way, the duration of the k-th note of onset $o_j^{(a)}$ (denoted as $\hat{d}_{j_k;a}$) in the set of upcoming onsets can be estimated as

$$\hat{d}_{j_k;a} = \mathrm{art}_i \cdot 2^{\hat{y}_k^{(\log \mathrm{art})}\left[o_j^{(a)}\right]} \cdot \mathrm{duration}\left(n_{j_k}^{(a)}\right) \cdot \mathrm{BP}_i, \tag{8.34}$$

where art_i is the moving average of the articulation ratio of the soloist computed using Equation (8.29).

Dynamics

The maximal MIDI velocity for each onset in $\boldsymbol{\mathcal{O}}_{next}^{(a)}\left(o_i^{(s)}\right)$ is given by

$$\hat{v}_{j;a}^{perf} = \mathrm{vel}_i \cdot \hat{\mathbf{y}}^{(\mathrm{vel}_{trend})}\left[o_j^{(a)}\right], \tag{8.35}$$

[6] Given our current understanding of general relativity, it is very unlikely that any accompaniment system would be able to do so.

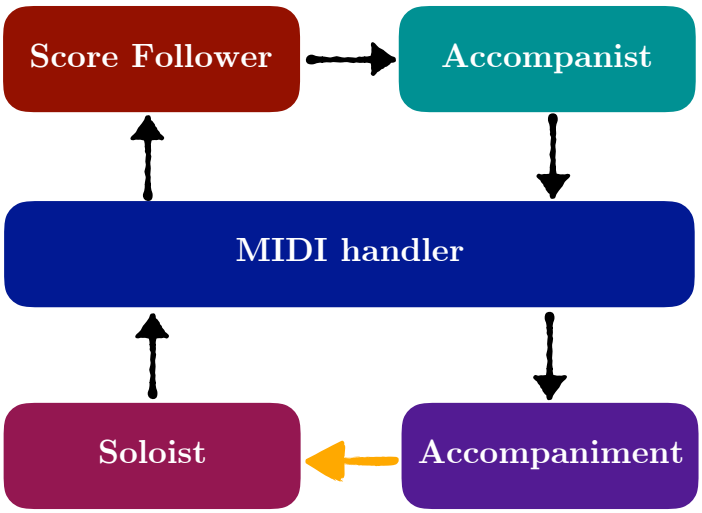


Figure 8.2: Schematic representation of the modules of the ACCompanion. Black arrows represent the flow of information for computing the accompaniment part. The thick orange arrow represents the implicit effect of the system output on the (live) soloist.

where vel_i is the moving average of the MIDI velocity of the soloist, computed using Equation (8.27). The MIDI velocity for the k-th note in accompaniment score onset $o_j^{(a)}$ is given by

$$\hat{v}_{j_k;a}^{perf} = \hat{v}_{j;a}^{perf} - \hat{y}_k^{(\text{vel}_{dev})}\left[o_j^{(a)}\right]. \tag{8.36}$$

Algorithm 8.1 summarizes the algorithm described in this section. The method `produce_midi` referenced in this algorithm is a MIDI messages scheduler, i.e. a method that receives information about the MIDI notes to be played, and sends the MIDI messages to the output MIDI device at the specified time. This method is the main interface between the accompanist and the MIDI Handler module of the ACCompanion.

8.4 The ACCompanion

This section provides a general overview of the ACCompanion. As described in Section 8.1, the system consists of three main components: a module that allows for score following (the SF), a module that adapts the expression of the accompaniment to the solo part (the accompanist), and a third module that handles the MIDI input and output (the MIDI handler). The relationship between these components is illustrated in Figure 8.2. In this figure, black arrows denote how the information flows from the soloist to each of the modules of the system. The orange arrow denotes the implicit effect of the system output on the (live) soloist[7].

[7]There is evidence that ensemble performers both deliberately and automatically adapt to each other's performance (Repp, 2001; van der Steen and Keller, 2013). It is important to note that this implicit effect of the output of the accompaniment system on the live soloist is not explicitly modeled in the current version of the ACCompanion.

Algorithm 8.1: `accompaniment_step`

Input:

- $o_i^{(s)}$: current matched solo score onset (with performed onset time $\hat{o}_{i,s}^{perf}$)
- BP_i: current estimated beat period of the solo part
- vel_i: moving average of the MIDI velocity of the last matched solo note
- art_i: moving average of the articulation ratio of the last matched solo note
- $\hat{\mathbf{y}}^{(\mathrm{vel}_{trend})}$: normalized predictions for MIDI velocity trend
- $\hat{\mathbf{y}}^{(\log \mathrm{bpr})}$: normalized predictions for $\log \mathrm{BPR}$
- $\hat{\mathbf{y}}^{(\mathrm{vel}_{dev})}$: normalized predictions for MIDI velocity deviations
- $\hat{\mathbf{y}}^{(\mathrm{tim})}$: normalized predictions for timing
- $\hat{\mathbf{y}}^{(\log \mathrm{art})}$: normalized predictions for $\log$ articulation

Get set of upcoming onsets in the accompaniment score

$$\boldsymbol{\mathcal{O}}_{next}^{(a)}(o_i^{(s)}) = \left\{ o_k^{(a)} \mid \mathrm{onset}\left(o_k^{(a)}\right) \geq \mathrm{onset}\left(o_i^{(s)}\right) \right\} \tag{8.37}$$

for $o_j^{(a)} \in \boldsymbol{\mathcal{O}}_{next}^{(a)}\left(o_i^{(s)}\right)$ **do**

Compute the average onset times

$$\hat{o}_{j;a}^{perf} = \begin{cases} \hat{o}_{i;s}^{perf} + 2^{\hat{\mathbf{y}}^{(\log \mathrm{bpr})}\left[o_{i_0}^{(a)}\right]} \cdot \mathrm{BP}_i \cdot \left(\mathrm{onset}\left(o_{i_0}^{(a)}\right) - \mathrm{onset}\left(o_i^{(s)}\right)\right) & \text{if } j = i_0 \\ \hat{o}_{j-1;a}^{perf} + 2^{\hat{\mathbf{y}}^{(\log \mathrm{bpr})}\left[o_j^{(a)}\right]} \cdot \mathrm{BP}_i \cdot \left(\mathrm{onset}\left(o_j^{(a)}\right) - \mathrm{onset}\left(o_{j-1}^{(a)}\right)\right) & \text{otherwise} \end{cases} \tag{8.38}$$

for $n_{j_k}^{(a)} \in o_j^{(a)}$ **do**

Compute onset time of note $n_{j_k}^{(a)}$

$$\hat{o}_{j_k;a}^{perf} = \max\left(\hat{o}_{i;s}^{perf}\,;\, \hat{o}_{j_k;a}^{perf} - \hat{y}_k^{(\mathrm{tim})}\left[o_j^{(a)}\right]\right) \tag{8.39}$$

Compute MIDI velocity of note $n_{j_k}^{(a)}$

$$\hat{v}_{j_k;a}^{perf} = \mathrm{vel}_i \cdot \hat{\mathbf{y}}^{(\mathrm{vel}_{trend})}\left[o_j^{(a)}\right] - \hat{y}_k^{(\mathrm{vel}_{dev})}\left[o_j^{(a)}\right] \tag{8.40}$$

Compute duration of note $n_{j_k}^{(a)}$

$$\hat{d}_{j_k;a} = \mathrm{art}_i \cdot 2^{\hat{y}_k^{(\log \mathrm{art})}\left[o_j^{(a)}\right]} \cdot \mathrm{duration}\left(n_{j_k}^{(a)}\right) \tag{8.41}$$

Send performance information to MIDI messages scheduler

`produce_midi` $\left(n_{j_k}^{(a)}, \hat{v}_{j_k;a}^{perf}, \hat{o}_{j_k;a}^{perf}, \hat{d}_{j_k;a}\right)$

The rest of this section is structured as follows: In Section 8.4.1 we discuss implementation details of the ACCompanion. Section 8.4.2 discusses two methods for live demos for visualizing and showcasing how the ACCompanion works. Section 8.4.3 describes the participation of the ACCompanion in a competition for computational accompaniment systems. Finally, Section 8.4.4 describes two videos produced to showcase the capabilities of the ACCompanion.

8.4.1 Implementation Details

As the case of the Basis Mixer described in Chapter 7, the ACCompanion is implemented in Python 2.7[8] and relies heavily on Numpy[9]. In contrast to the Basis Mixer, which renders the performance of a score in an *offline* fashion, for the purpose of expressive accompaniment it is important that the system runs in real-time. Furthermore, as previously mentioned, each of the three modules of the ACCompanion has to run in parallel with each other. We solve this issue by using Multiprocessing, a Python module that allows for spawning processes in parallel[10].

The implementation of the KHMM in the SF module is based on the HMM implemented in madmom[11], a Python library for audio signal processing which focuses on music information retrieval (MIR) tasks.

The MIDI handler uses the Python-RTMidi library for handling input and output of MIDI messages in real-time[12]. This module includes functions for selecting the input and output MIDI ports. The incoming MIDI messages from the solo part are sent to the SF module. The MIDI handler includes the `produce_midi` method referenced in Algorithm 8.1. This method is a MIDI messages scheduler that receives information about the MIDI notes to be played, and sends the MIDI messages to the output MIDI device at the specified time. The messages are stored in a queue, which is updated every time that the score follower identifies a performed note as being a note in the solo score. For the actual implementation of the system, `produce_midi` also includes a way to identify whether a note in the accompaniment has already been played, so that there are no duplicate notes.

Algorithm 8.2 presents a schematic description of the ACCompanion. The method `perf_info` referenced in this algorithm encapsulates the estimation of the local playing style of the soloist in terms of the beat period BP_i , the moving average of the MIDI velocity of the solo, and the moving average of the articulation ratio of the solo, as described in Section 8.3.2.

8.4.2 Methods for Demonstrating and Visualizing the ACCompanion

In this section we describe two methods for illustrating how the ACCompanion works. The first of these methods is a visualization of the performance of the solo and accompaniment parts, and the second method is the use of a controller to exaggerate aspects of the performance of the accompaniment part.

In addition to the modules described above, the current implementation of the ACCompanion includes a display that provides visual feedback of the performance of a piece. We refer to this visualization tool as the *performance visualizer* of the ACCompanion. This visualization displays both the solo and accompaniment parts in a piano roll in real time. The performance visualizer

[8] https://www.python.org.
[9] http://www.numpy.org.
[10] https://docs.python.org/2/library/multiprocessing.html.
[11] https://madmom.readthedocs.io/en/latest/.
[12] https://spotlightkid.github.io/python-rtmidi/.

Algorithm 8.2: ACCompanion v. 0.1

Data:

- $\mathfrak{S}^{(a)}$: Score of the accompaniment part
- $\mathfrak{S}^{(s)}$: Score of the solo part
- $\hat{\mathbf{y}}^{(\text{vel}_{trend})}$: normalized predictions for MIDI velocity trend
- $\hat{\mathbf{y}}^{(\log \text{bpr})}$: normalized predictions for $\log \text{BPR}$
- $\hat{\mathbf{y}}^{(\text{vel}_{dev})}$: normalized predictions for MIDI velocity deviations
- $\hat{\mathbf{y}}^{(\text{tim})}$: normalized predictions for timing
- $\hat{\mathbf{y}}^{(\log \text{art})}$: normalized predictions for $\log$ articulation
- BP_{init}: Initial beat period
- vel_{init}: Initial MIDI velocity trend
- art_{init}: Initial articulation ratio

while *Performance of* $\mathfrak{S}$ **do**

  if $\min$ *onset* $\left(\boldsymbol{\mathcal{O}}^{(a)}\right) \leq \min$ *onset* $\left(\boldsymbol{\mathcal{O}}^{(s)}\right)$ **then**

    Start accompaniment

$$\texttt{accompaniment\_step}\left(o_0^{(a)}, \text{BP}_{init}, \text{vel}_{init}, \text{art}_{init}\right)$$

  if *Soloist plays note* $n^{(p)}$ **then**

    Estimate if note belongs to the score position or if it an insertion using the KHHM:

$$o_{est} = \text{sfollower}(n^{(p)}) \tag{8.42}$$

    if o_{est} *is not an insertion* **then**

      Assign to a note in the score:

$$o_i^{(s)} = o_{est} \tag{8.43}$$

      Get performance parameters of the solo:

$$\text{BP}_i, \text{vel}_i, \text{art}_i = \texttt{perf\_info}\left(o_i^{(s)}\right) \tag{8.44}$$

      Update accompaniment:

$$\texttt{accompaniment\_step}\left(o_i^{(s)}, \text{BP}_i, \text{vel}_i, \text{art}_i\right)$$

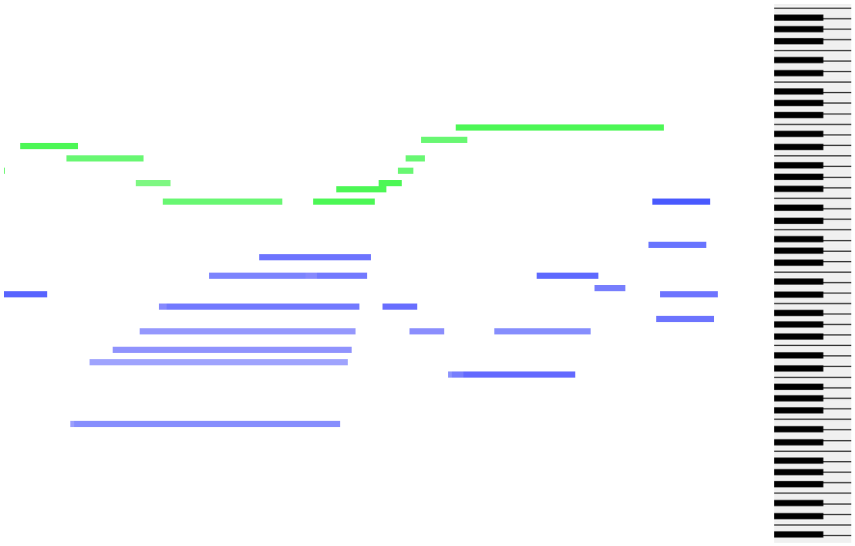


Figure 8.3: Screenshot of the performance visualizer of the ACCompanion.

was designed and implemented by Martin Bonev under supervision from Werner Goebl, Laura Bishop and myself. A simple color scheme is used to illustrate the loudness, with brighter color lines representing louder notes. Solo and accompaniment parts are distinguished using different colors. In the solo part, green lines represent correctly played notes, while inserted and misplayed notes are drawn in red. The accompaniment part is presented with purple lines. A screenshot of the visualization is shown in Figure 8.3.

For the demonstration purposes, we use a USB knob controller, a PowerMate by Griffin Technology[13], to exaggerate or minimize certain aspects of the expressive performance by controlling the scaling of the expressive parameters in real time. In the current implementation, the controller can scale the overall articulation, as well as the note-wise deviations of MIDI velocity. The angle of rotation of the PowerMate is mapped to a scalar variable pm that lies between -1 to 1, which we refer to as the value of the PowerMate. Rotating the PowerMate to the right increases the value of pm until it reaches 1 and then does not increase further even if the user keep rotating right. Rotating the PowerMate to the left decreases the value of pm until it reaches -1. Pressing the button of the PowerMate resets pm $= 0$. Using this parameter, the PowerMate exaggerates the spread of MIDI velocities and the articulation by scaling the normalized predictions, i.e.,

$$\hat{\mathbf{y}}^{(\text{vel}_{dev})} \leftarrow \alpha^{\text{pm}_\tau} \hat{\mathbf{y}}^{(\text{vel}_{dev})} \tag{8.45}$$

$$\hat{\mathbf{y}}^{(\log \text{art})} \leftarrow \alpha^{\text{pm}_\tau} \hat{\mathbf{y}}^{(\log \text{art})} \tag{8.46}$$

where α is a constant (in the current implementation $\alpha = 3$) and pm_τ is the value of the PowerMate at time τ. For practical reasons, it would be very resource-consuming to rescale the normalized predictions every time that the value of the PowerMate changes, since this would involve calling the `accompaniment_step` method and updating the scheduler of output MIDI messages. Instead, we only update the value of the normalize parameters every time that there is a note performed by the soloist that is matched to the solo score.

[13] https://griffintechnology.com/us/products/stylus-keyboards/powermate

8.4.3 AccompaniX 2017

In May 2017, an early version of the ACCompanion participated in the AccompaniX 2017: Artificial Duet Performer Competition[14]. This competition aimed to showcase systems capable of producing expressive accompaniment in coordination with a human soloist. The most successful systems would be able to adapt in real-time to variabilities in timing, dynamics and articulation in the solo part. Competition participants were given scores and pre-recorded soloist performances for four pieces, all containing monophonic solo lines and polyphonic accompaniment. The task was then to produce a coordinated accompaniment performance for one of the pieces, with the constraint that the system producing accompaniment had to be causal – that is, it could not make use of future information from the human performance, even when running off-line (i.e. with a pre-recorded performance, not in real-time). Participants of the competition had to upload their systems to a server before the competition organizers used them to produce performances for evaluation, which meant that we as participants were unable to adjust parameters to the specific case of the performance. A limitation of the competition format was the human soloist performances were pre-recorded, so interaction between duet partners was only one-way (the system could adapt to the soloist, but not vice versa).

An evaluation of submitted systems was conducted through an online survey in which listeners rated the quality of the duet performances.

The ACCompanion won a U.S. $500 Award for Creative Achievement. Chris Raphael's Music Plus One (Raphael, 2001a, 2010) won the first prize.

8.4.4 Videos for ISMIR 2017

We produced two videos showcasing the capabilities of the system to be presented at the Late Breaking/Demo session of the 18th International Society for Music Information Retrieval Conference held in Suzhou, China[15]. These videos were recorded at the Fanny Hensel-Mendelssohn Hall at the University of Music and Performing Arts Vienna, in collaboration with Werner Goebl, Gerhard Widmer, Laura Bishop, and Martin Bonev. The pieces were performed on a Bösendorfer CEUS 280VC grand piano, a computer-controlled piano that allows for capturing the performance information in MIDI format.

Figure 8.5 shows the setup for recording the demonstration videos. The audio of the piano was captured using two microphones. The accompaniment part was generated using the Bönsedorfer preset of the ESX24 instrument sampler included in Logic Pro X[16].

In these demo videos, the piano roll (i.e. the performance visualization described above) shows the notes performed by the soloist in green and the accompaniment in purple. Wrong notes performed by the soloist are displayed in red. The intensity of the colors in the piano roll corresponds to the loudness (the MIDI velocity) of the performed notes. The performance errors by the soloists are deliberate to showcase the capabilities of the system.

In the first video, Werner Goebl performs the right-hand part of the second movement of the Sonata No. 16 KV. 545 by W. A. Mozart. The left-hand part is performed by the ACCompanion. In the second video, Gerhard Widmer performs The Wild Geese, the piece that was submitted for the AccompaniX 2017 competition. Note that in these performances there are deliberate

[14] http://bregman.dartmouth.edu/turingtests/Automatic-accompaniment.
[15] These videos are available online: http://www.carloscancinochacon.com/documents/online_extras/ismir2017lbd/online_extras.html.
[16] https://www.apple.com/logic-pro/plugins-and-sounds/.

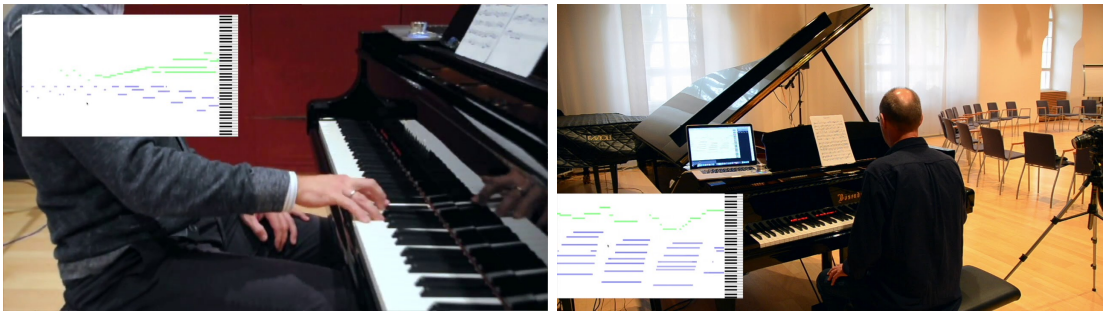


Figure 8.4: Thumbnails of the videos demonstrating the capabilities of the ACCompanion. On the left: Werner Goebl performing the right hand part of second movement of the Sonata No. 16 KV. 545 by W. A. Mozart. On the right: Gerhard Widmer performs the solo part of The Wild Geese. The piano rolls represent the visualization described in Section 8.4. These videos were recorded at the Fanny Hensel-Mendelssohn Hall at the University of Music and Performing Arts Vienna. The piano is a Bösendorfer CEUS 280VC.

attempts by the soloist to play incorrect and/or repeated notes (especially in the second part of the second movement of Mozart's Sonata No. 16) to showcase the capabilities and robustness of the system.

8.5 Conclusions

In this chapter, we have presented a first prototype of an automatic accompaniment system. The current version of the ACCompanion uses an HMM-based monophonic score follower to match the input of the soloist to a position in the score, combined with an adapted version of the BM to generate an expressive accompaniment part that reacts to the performance style of the soloist.

Future versions of the ACCompanion need to improve in two main directions: better score following (including expanding the system to follow complex polyphonic music) and better generation of expressive accompaniment parts.

For the first problem, it would be interesting to test more sophisticated HMM-based approaches (including second order HMMs and hidden semi-Markov Models) following the work by Nakamura et al. (2015a). Another interesting route could be through the use of on reinforcement learning, following recent promising work by Dorfer et al. (2018).

For the second problem, we aim at integrating more complex variants of the BM framework for expressive performance, trained on real ensemble performance data. In particular, it would be important to develop a model that uses the performance of the soloist as an input to make predictions of the accompaniment part, instead of simply scaling and centering a pre-computed performance, as is the case of the current implementation. Additionally, it would be interesting to expand the model to also predict expressive nuances in the soloist performance, instead of only reacting to them. In this way, the estimation of the local playing style of the soloist could be *predictive* rather than *reactive*. A model capable of predicting expression would likely be more engaging to perform with than a purely reactive system. Preliminary tests showed that the experience of the performer is very dependent on the accuracy of the score following, in particular to the estimation of tempo. It would be interesting to explore other possibilities for tempo models including more sophisticated sensorimotor synchronization models such as the

Figure 8.5: Setup for recording the demonstration videos at of the ACCompanion. In the picture (left) Gerhard Widmer, Werner Goebl (at the piano) and myself (in front of the computer).

.

adaptation and anticipation model by van der Steen and Keller (2013).

For this early prototype, we have not yet done any thorough evaluation, other than playing with it in order to obtain a general impression. Later versions will, of course, be evaluated in much more systematic ways, for instance by measuring how well their performances correlate with those of human musicians. However, this quantitative approach to evaluation is intrinsically problematic, as it makes very limiting assumptions regarding what kinds of performances are musically meaningful and 'good'. Ultimately, really meaningful tests will have to involve the judgment of human musicians and listeners. We defer this to later stages of the project, where we hope to have a more complete and musically sophisticated system.

9 In Conclusion: A Comprehensive and Critical Review of the Current State of the Field

By way of conclusion, this final chapter offers a comprehensive survey and critical review of the current state of the art of the broad field of computational performance modeling, written after, and including, my own research as presented in this thesis. This text will be published as

- Cancino-Chacón, C., Grachten, M., Goebl, W., and Widmer, G. (2018). Computational Models of Expressive Music Performance: A Comprehensive and Critical Review. *To appear in Frontiers in Digital Humanities*

The text of the article is reproduced here "as is", including the original introduction and motivation sections, so that it can be read as a self-contained document, independently of the rest of this thesis.

9.1 Introduction

The way a piece of music is performed is a very important factor influencing our enjoyment of music. In many kinds of music, particularly Western art music, a good performance is expected to be more than an exact acoustic rendering of the notes in the score. Performers have certain liberties in shaping various parameters (e.g., tempo, timing, dynamics, intonation, articulation, etc.) in ways that are not prescribed by the notated score, and are expected to use these to produce an *expressive* rendition of the piece in question. This applies not only to classical music, where interpretation and performance are perpetual topics of artistic and aesthetic discussion, but to virtually all kinds of music. *Expressive performance*, as we will call it in the following, is known to serve several purposes: foremost, to express and communicate the performer's understanding of structure and affective content ('meaning') inherent in a composition, and in this way to bring out dramatic, affective, and emotional qualities that, in the best case, may engage and affect the listeners emotionally. Expert musicians learn (mostly implicit) performance rules through many years of focused and intensive practice and intellectual engagement with music. Given the central importance of this subtle art, the principles behind, and processes involved in, expressive performance should be a central topic of research in music and music psychology.

The systematic study of expressive music performance is a relatively young field, starting in the first half of the 20th Century with the advent of recording technology (Binet and Courtier, 1896; Seashore, 1938). The second half of the 20th Century saw an increased interest in looking at performance from the perspective of music psychology and cognition (Clynes, 1969; Gabrielsson, 1974; Longuet-Higgins and Lee, 1982, 1984; Clynes, 1986, 1987; Palmer, 1996). The field gained more attraction in the late 1980's, with advances in computers and electronic instruments, which facilitated more precise data capturing (Kirke and Miranda, 2013). Music performance science is a highly inter-disciplinary field, and a thorough review of the state of the art of the full field is outside the scope of this chapter. We refer the interested reader to the very comprehensive review articles by Palmer (1997) and Gabrielsson (1999, 2003). For a review of performance

research from a musicological point of view see Rink (1995, 2002, 2003). For philosophical perspectives on expressiveness in music, we refer the reader to Davies (1994, 2001).

This chapter focuses on a narrower and more specific topic: *computational models of expressive performance*, that is, attempts at codifying hypotheses about expressive performance – as mappings from score to actual performance – in such a precise way that they can be implemented as computer programs and evaluated in systematic and quantitative ways. This has developed into a veritable research field of its own over the past two decades, and indeed the present work is not the first survey of its kind; previous reviews of computational performance modeling have been presented by Widmer and Goebl (2004), De Poli (2004), and Kirke and Miranda (2013).

The new review we offer here goes beyond these earlier works in several ways. In addition to providing a comprehensive update on newer developments, it is somewhat broader, covering also semi-automatic and accompaniment systems, and discusses the components of the models in more detail than previous reviews. In particular, it provides an extended critical discussion of issues involved in model choices – particularly the selection and encoding of input features (score representations) and output parameters (expressive performance dimensions) – and the evaluation of such models, and from this derives some research directions that should be pursued with priority, in order to advance the field and our understanding of expressive music performance. As in earlier papers, we focus on models for notated music, i.e., music for which a musical score (a symbolic description of the music) exists. This includes most Western art music. A review of models of expressive performance for non-western or improvised music traditions is outside the scope of this work.

The rest of this text is organized as follows: Section 9.2 introduces the concept of computational music performance models, including possible motivations, goals, and general model structure. Section 9.3 attempts to give a comprehensive overview of the current state of the art, focusing on several current trends in the field. Section 9.4 offers a critical discussion of crucial modeling aspects, and offers a critical view on the ways in which performance models are currently evaluated. Section 9.5 concludes the chapter with a list of recommendations for future research.

9.2 Computational Modeling of Expressive Performance

9.2.1 Motivations for Computational Modeling

Formal and computational models of expressive performance are a topic of interest and research for a variety of scientific and artistic disciplines, including computer science, music psychology and musicology, among others. Accordingly, there is an wide variety of motivations for this kind of modeling. Broadly speaking, we can categorize these motivations into two groups: on the one hand, computational models can be used as an analytical tool for understanding the way humans perform music; on the other hand, we can use these models to generate (synthesize) new performances of musical pieces in a wide variety of contexts.

As analysis tools, computational models permit us to study the way humans perform music by investigating the relationship between certain aspects of the music, like the phrase structure, and aspects of expressive performance, such as expressive timing and dynamics. Furthermore, they allow us to investigate the close relationship between the roles of the composer, the performer and the listener (Kendall and Carterette, 1990; Gingras et al., 2016). Expressive performance and music perception form a feedback loop in which expressive performance actions (like a slowing down at the end of a phrase) are informed by perceptual constraints or expectations, and the perception of certain musical constructs (like grouping structure) is informed by the

way the music is performed (Chew, 2016). In this way, computational models could also be used to enhance our understanding of the way humans listen to music.

On the other hand, computational performance models can be interesting in their own right, as tools for generating automatic or semi-automatic performances. In this case, a generative system might attempt to produce a *convincing* or *human-like* performance of a piece of music given its score (Friberg et al., 2006; Grachten and Widmer, 2012; Okumura et al., 2014) or try to play alongside human musicians, not only tracking their expressive performance but also introducing its own expressive nuances (Xia et al., 2015; Cancino-Chacón et al., 2017a). Such systems might have many applications, including realistic playback in music typesetting tools (such as Finale or MuseScore) and automatic expressive accompaniment for rehearsing. Also, there is now a renewed interest in systems that automatically generate (i.e. compose) music. As pointed out by Herremans et al. (2017), automatic performance systems might be an important component in making automatic music generation usable by the general public.

From a philosophical perspective, the idea of musical expressiveness presents a number of issues (Davies, 2001). Among these is the fundamental question of whether an expressive performance *can be fully captured* using numerical descriptors. For example, Bergeron and Lopes (2009) discuss whether a complete sonic description of the music without any visual component can fully convey the expressivity of music. That hearing *and* seeing a musical performance provides for a richer experience[1] is an interesting and plausible hypothesis, but this question goes beyond the scope of the present chapter. In any case, it should be undisputed that there *is* more than enough expressivity to be perceived – and thus also modeled – from just a sonic representation; after all, listening to a recorded performance is still the predominant way of enjoying music, and it can be a rewarding experience.

9.2.2 Components of the Performance Process

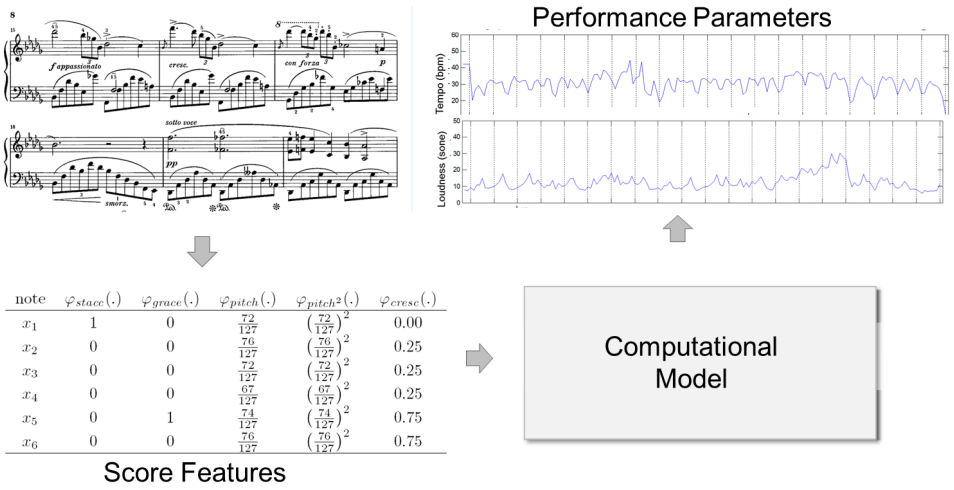
In her seminal review, Palmer (1997) groups the reported work in three sections that can be taken to reflect the human cognitive processes involved in performing a piece of notated music:

1. **Interpretation**. According to Kendall and Carterette (1990), music performance is a communication process in which information (emotional and semantic content of the piece) flows from the composer to the performer to the listener. We note here that these should be regarded as roles rather than agents, since, for example, the composer and performer may be embodied by the same person. An important task for the performer is to determine how to convey the message from the composer to the listener. Palmer refers to *interpretation* as the act of arriving at a conceptual understanding of structure and emotional content or character of a given piece, in view of a planned performance. Examples of relevant structural aspects are the grouping and segmentation of sequences into smaller subsequences to form hierarchical levels – such as those proposed by Lerdahl and Jackendoff (1983) in their Generative Theory of Tonal Music (GTTM).

2. **Planning**. Through planning the performer decides how to relate the syntax of musical structure to expression through style-specific actions and constraints. Such actions include, e.g., the use of arch-like patterns in dynamics and tempo to elucidate the phrasing structure (Todd, 1992; Friberg and Sundberg, 1999).

3. **Movement**. Finally, a performer needs to transform a performance plan into a concrete execution of the piece by means of physical movement. These movements can be seen as

[1] Or, as Bergeron and Lopes (2009) (quoting Robert Schumann) put it: "if Liszt played behind a screen, a great deal of poetry would be lost."



note	$\varphi_{stacc}(.)$	$\varphi_{grace}(.)$	$\varphi_{pitch}(.)$	$\varphi_{pitch^2}(.)$	$\varphi_{cresc}(.)$
x_1	1	0	$\frac{72}{127}$	$\left(\frac{72}{127}\right)^2$	0.00
x_2	0	0	$\frac{76}{127}$	$\left(\frac{76}{127}\right)^2$	0.25
x_3	0	0	$\frac{72}{127}$	$\left(\frac{72}{127}\right)^2$	0.25
x_4	0	0	$\frac{67}{127}$	$\left(\frac{67}{127}\right)^2$	0.25
x_5	0	1	$\frac{74}{127}$	$\left(\frac{74}{127}\right)^2$	0.75
x_6	0	0	$\frac{76}{127}$	$\left(\frac{76}{127}\right)^2$	0.75

Figure 9.1: Computational Modeling Scenario.

embodied human–music interactions which have an impact on the way humans perform and perceive music (Leman et al., 2017a).

In Section 9.4.2, we will present a discussion on how different aspects and methods of computational modeling of performance fit into these categories, as well as the implications of the choice for the modeling.

9.2.3 Components of Computational Models

Ideally, a full computational model of expressive performance should cover all three of the above aspects. However, the models described in the literature so far focus almost exclusively on the *planning* process, conceptualizing it as a *mapping* from a given score to specific patterns in various performance parameters (e.g., timing or dynamics) and, eventually, to an acoustic realization of the piece (De Poli, 2004). Thus, in the remainder of this review we will adopt this (admittedly too) limited view and discuss existing performance models in this context.

Kirke and Miranda (2013) proposed a generic framework for describing research in expressive performance. In the present chapter, we adopt a simplified version of this framework involving three main components by which computational performance models (in the limited sense as explained above) can be characterized. The resulting simple modeling scenario is shown in Figure 9.1, along with a fragment of a musical score.

By *score features* – which are the inputs to the computational model – we denote descriptors used to represent a piece of notated music. Some of these features may be given directly by the score (such as notated pitches and durations), while others may be computed from the score in more or less elaborate ways, by some well-defined procedure (such as the cognitive features discussed in Section 9.3.3). Features can range from low-level descriptors such as (MIDI) pitches (Grindlay and Helmbold, 2006; Friberg et al., 2006; Cancino Chacón and Grachten, 2015) and hand-crafted features, like encodings of metrical strength (Grindlay and Helmbold, 2006; Giraldo and Ramírez, 2016a); to cognitively inspired features, like Narmour's Implication-Realization (IR) descriptors (Flossmann et al., 2013; Giraldo and Ramírez, 2016b), or even features learned directly from the score using unsupervised machine learning (Grachten and Krebs, 2014; van Herwaarden et al., 2014). The process of extracting score features corresponds to the *Music/Analysis* module in Kirke and Miranda (2013)'s framework and can be seen as at least partly related to Palmer's *Interpretation* aspect (see above).

An *expressive parameter* – the output of a model – is a numerical encoding of an aspect of expressive performance. Since most systems deal with piano music, the most common descriptors relate to loudness, expressive tempo and timing, and articulation (Kirke and Miranda, 2013; Widmer and Goebl, 2004), but of course they can also include other parameters like timbral features (Raphael, 2009; Ohishi et al., 2014) and intonation (Clynes, 2005), or higher-level patterns such as "pulse microstructure" (Clynes, 1987). The expressive parameters correspond to the outputs of the *Performance Knowledge* module in Kirke and Miranda (2013)'s framework. Section 9.4.1 presents a critical review of the choices involved in selecting and encoding these parameters.

A *computational model* then, in our context, is any computable function that maps score features to expressive parameters or, to be more precise, can make a *prediction* of the values of expressive parameters, given a score (represented via score features) as input. In music performance modeling, this is typically done by means of mathematical functions (probabilistic models, artificial neural networks, etc.) (Teramura et al., 2008; Kim et al., 2010; Grachten and Widmer, 2012) or by means of rules (Friberg et al., 2006; Canazza et al., 2015). Some of these models can be *trained* using a dataset of expressive performances. The model/function corresponds to the *Performance Knowledge* and the *Performance Context* in Kirke and Miranda (2013)'s framework; the training of the model corresponds to the *Adaptation Process*, and the datasets are the *Performance Examples*.

9.3 A Synopsis of Current State and Recent Trends

In this section we discuss some of the recent trends in computational performance modeling. This brief overview is meant as an update to earlier review papers by Widmer and Goebl (2004), De Poli (2004) and Kirke and Miranda (2013). In the following, we will refer to a model as *static* if its predictions only depend on a single event in time (e.g. linear regression, feed forward neural networks), and *dynamic* if its predictions can account for time-dependent changes (e.g. hidden Markov models or recurrent neural networks).

9.3.1 Data-driven Methods for Analysis and Generation of Expressive Performances

A first noteworthy trend in recent research is an increasing focus on *data-driven* approaches to performance modeling, relying on *machine learning* to infer score-performance mappings (and even the input score features themselves) from large collections of real data (scores and performances). This is in contrast to *rule-based* approaches where performance rules are manually designed, based on musical hypotheses.

An important example of the rule-based variety is the KTH model (Sundberg et al., 1983; Friberg et al., 2006), developed at the Royal Institute of Technology (KTH) in Stockholm. These rules were developed and evaluated through an iterative *analysis-by-synthesis* approach involving judgments by experts and listening tests. A performance is shaped by a (linear) combination of the effects of the rules, which the user can weigh individually. The KTH model has been implemented as a software package called *Director Musices* (DM) (Friberg et al., 2000; Masko et al., 2014). Recent versions of the KTH model include cognitively motivated rules regarding musical accents (Bisesi et al., 2011). Friberg and Bisesi (2014) study the use of the system for modeling stylistic variations for Baroque, Romantic and Contemporary art music. The KTH model won the first prize at the RenCon, a competition for computational models of performance, in 2004 (see also Section 9.4.3 below).

While early data-driven approaches (Widmer, 1995, 1996, 2000; Widmer and Tobudic, 2002; Widmer, 2003) aimed at learning explicit performance rules at various structural levels (from individual notes to higher phrasing levels), using methods like instance-based learning and inductive logic programming, recent advances in machine learning – in particular relating to probabilistic graphical models and (deep) neural networks – have led to a surge of such methods in computational performance modeling, which will be reviewed in the following.

Data-driven Methods for Performance Analysis

As tools for analysis, computational methods can be used for several purposes, including studying the relationship between structural aspects of the score and specific aspects of a performance, or for comparing expressive renderings by different performers.

Explaining/modeling aspects of performance An important question in analyzing expressive performance is determining the likely 'causes' of observed performance patterns, i.e, structural or other aspects of a piece that would 'explain' why a certain passage was (or needs to be) played in a certain way. By analyzing large amounts of data, data-driven methods can find systematic relations between measured performance aspects (e.g., changes in tempo and dynamics) and various structural aspects of a musical score (e.g., pitch content, metrical and phrasing structure), notated performance indications (e.g., dynamics markings such as ***piano*** and ***forte*** or articulation marks such as ***legato*** slurs and ***staccato***), or even aspects related to our perception of music, like melodic expectation (as far as we are able to infer or compute these in a reliable way from a score).

Examples of such approaches include the work by Kosta et al. (2014, 2015, 2016), who focus on the relationship between dynamics markings and expressive dynamics, and the *Basis Function Model* (Grachten and Widmer, 2012) – a framework that encodes score properties via so-called basis functions – which attempts to quantify the contribution of a variety of score descriptors (such as pitch, metrical position and dynamics markings) to expressive dynamics (Cancino-Chacón et al., 2017d) and timing (Grachten and Cancino-Chacón, 2017). Fu et al. (2015) study timing deviations in arpeggiated chords with statistical methods. Gingras et al. (2016) and Cancino-Chacón et al. (2017c) focus on linking information-theoretic features quantifying the expectation of musical events in listeners, to expressive timing. Caramiaux et al. (2017) study performers' skill levels through variability in timing and features describing finger motion. Marchini et al. (2014) study the use of score features describing horizontal (i.e. melodic) and vertical (i.e. harmonic) contexts for modeling dynamics, articulation and timbral characteristics of expressive ensemble performances, focusing on string quartets. Using machine learning and feature selection techniques, Giraldo and Ramírez (2016b) and Bantula et al. (2016) evaluate a number of score descriptors in modeling expressive performance actions for jazz guitar and jazz ensembles, respectively.

A second form of analysis focuses on specific patterns and characteristics in curves of expressive parameters. This includes work on methods for visualizing expressive parameters and their characteristics (Langner and Goebl, 2003; Grachten et al., 2009; Chew and Callender, 2013), on inferring performance strategies like phrasing from expressive timing (Chuan and Chew, 2007) or dynamics (Cheng and Chew, 2008), and clustering of patterns of (phrase-level) tempo variations (Li et al., 2014, 2015, 2016, 2017). The results obtained with such methods support the existence of common performance strategies (Cheng and Chew, 2008; Kosta et al., 2016; Li et al., 2014). Quantitative studies on the contribution of various score features to expressive parameters reveal well-known relationships, like the importance of pitch (height) for predicting expressive

dynamics, and the relationship between metrical features and timing deviations. At the same time, some results indicate that aspects of performance (like expressive tempo and dynamics) might be related in more than one way to structural aspects of the music, e.g. phrasing has been shown to be related to dynamics (Cheng and Chew, 2008) or timing (Chuan and Chew, 2007). An interesting finding is the importance of features and models that allow for describing the musical contexts, both horizontal (temporal) (Kosta et al., 2016; Gingras et al., 2016; Grachten and Cancino-Chacón, 2017) and vertical (i.e. harmonic) (Marchini et al., 2014).

Comparing expressive performances A different approach to analyzing expressive performances is to compare different renditions of the same piece by different performers, which allows for studying of commonalities and differences in performance strategies. Some of the work in this direction follows an unsupervised approach, which does without any score information, instead focusing on comparing aligned curves of expressive parameters that encode the performances. Sapp (2007, 2008) presents a graphical approach and explores different metrics for comparing collections of performances of the same piece. Liem et al. (2011) and Liem and Hanjalic (2011) propose a method for comparing expressive timing by studying alignment patterns between expressive performances of the same piece using standard deviations and entropy, respectively. Liem and Hanjalic (2015) use Principal Components Analysis (PCA) to localize areas of cross-performance variation, and to determine similarities between performances in orchestral recordings. Peperkamp et al. (2017) present a formalization of relative tempo variations that considers performances as compositions of functions which map performance times to relevant feature spaces. Rather than focusing on a single aspect like dynamics or timing, Liebman et al. (2012) present a phylogenetic approach that compares and relates performances of two solo violin pieces by different performers, using performance descriptors like bowing, tempo changes, and phrase duration. Grachten et al. (2017) use Basis Function models to assess the contribution of score features pertaining to individual orchestral instruments to the overall loudness curves, using differential sensitivity analysis, which allows for graphically comparing pairs of recordings of the same piece by different conductors and orchestras. Methods for comparing performances can be used for identifying musicians by their individual performance styles. This has been demonstrated for violinists (Molina-Solana et al., 2008, 2010a), saxophone players (Ramírez et al., 2007), and pianists (Stamatatos and Widmer, 2005; Saunders et al., 2008; Grachten and Widmer, 2009; Molina-Solana et al., 2010b).

Computational methods for performance comparison have produced some interesting results. They support the idea of common performance strategies across performers, as well as consistent individual differences between performers. Furthermore, they seem to support musicologically plausible hypotheses such as the change in playing style over the years and differences between mainstream and historically informed performance styles, while only providing weak evidence for the existence of "performance schools" (Liebman et al., 2012). The formalization of tempo proposed by Peperkamp et al. (2017) provides an interesting mathematical constraint on tempo curves as convex linear combinations of tempo variation functions.

In spite of all this, there has been only little progress in really understanding the way humans perform music expressively. An important issue is that effectively all studies are limited to small datasets (at least compared to other machine learning domains) that only contain a small selection of pieces and/or performers. This raises the question how well (or if) the insights gained from these studies generalize to other performers or kinds of music. Also, most models rely on features that capture only small local contexts, so that the resulting models cannot properly account for long temporal dependencies that might be important for understanding global aspects of performance expression. We still largely fail to understand how to model long-term, non-contiguous relationships in complex music. The hope is that recent advances in

(deep) machine learning may open new possibilities here (Widmer, 2017).

Data-driven Methods for Performance Generation

In this section we examine recent work on autonomous generative systems. While computational methods for analysis tend to focus on explaining a single aspect of expression, generative models most commonly have to consider more expressive parameters, with expressive tempo/timing and dynamics being the most popular. As previously discussed, a major trend in this area is the use of complex probabilistic approaches and the use of neural-network-based methods.

Probabilistic Approaches In a nutshell, probabilistic approaches describe expressive performances by modeling the probability distribution of the expressive parameters given the input score features. Table 9.1 presents some of the recent probabilistic performance systems in terms of their computational models, expressive parameters, and score features. Please note that the column relating to score features in Table 9.1 is not exhaustive, given the potentially large number of features used in each model.

While each model conceptualizes a music score and its corresponding expressive performance differently, there are some interesting commonalities. Several researchers use variants of Hidden Markov Models (HMMs) to describe the temporal evolution of a performance, such as Hierarchical HMMs (Grindlay and Helmbold, 2006), Dynamic Bayesian Networks (DBNs) (Widmer et al., 2009; Flossmann et al., 2011, 2013), Conditional Random Fields (CRFs) (Kim et al., 2010, 2011, 2013) or Switching Kalman Filters (Gu and Raphael, 2012). Furthermore, most models assume that the underlying probability distribution of the expressive parameters is Gaussian (Grindlay and Helmbold, 2006; Teramura et al., 2008; Gu and Raphael, 2012; Flossmann et al., 2013; Okumura et al., 2014). A different approach is taken by Kim et al. (2013) and Moulieras and Pachet (2016), who use maximum entropy models to approximate the underlying probability distributions. While most models focus on Western classical music, Moulieras and Pachet (2016) focus on expressive performance of jazz piano.

In terms of expressive parameters, most models describe expressive dynamics using the note-wise MIDI velocity. This is mostly done by either making predictions from a static model (Teramura et al., 2008), focusing only on monophonic melodies (Grindlay and Helmbold, 2006; Gu and Raphael, 2012; Moulieras and Pachet, 2016), or assuming a decomposition of the piece into monophonic streams (Kim et al., 2013; Okumura et al., 2014). On the other hand, there seems to be a variety of descriptors for expressive tempo and timing, with some models focusing on the *inter-beat interval* (IBI; a local estimation of the time between consecutive beats) or inter-onset interval (IOI; the time interval between consecutive onsets), some on the local *beats per minute* (bpm; the inverse of the IBI). Other models target local changes in their expressive parameters, by means of modeling their first differences[2]. Most models use a combination of low-level features – pitch, onset and duration of notes, as well as encodings of dynamics and articulation markings – and high-level features describing musically meaningful structures, such as metrical strength. Most systems only model expressive parameters independently, and the few exceptions focus on specific combinations of parameters, such as the ESP system (Grindlay and Helmbold, 2006) that jointly models tempo and tempo changes, but describes dynamics independently, and the model by Moulieras and Pachet (2016), which jointly models timing and dynamics.

[2]For a parameter p_i, the first (finite) difference refers to $\Delta p_i = p_i - p_{i-1}$.

Table 9.1: Probabilistic Models for Performance Generation

System	Computational Model	Expressive Parameters	Score Features
ESP Grindlay and Helmbold (2006)	Hierarchical HMMs	• Tempo: log IBI ratio and its first difference. • Dynamics: MIDI velocity	• Low-level: melodic interval. • High-level: metrical hierarchies and phrase structure (annotated)
NAIST Teramura et al. (2008)	Gaussian Processes	• Dynamics: MIDI velocity, • Timing: onset deviations (bpm) • Articulation: offset deviations (bpm)	• Low-level: pitch, duration, dynamics markings. • High-level: time signature, melody, relative pitch.
YQX Flossmann et al. (2011)	DBNs (+ learned articulation rules from Widmer (2003))	• Tempo: low frequency components of the log-IOI. • Timing: high-frequency components of log-IOI. • Articulation: duration ratio • Dynamics: log-MIDI velocity.	• Low-level: Pitch, contour. • High-level: Narmour's IR features, harmonic consonance
Gu & Raphael Gu and Raphael (2012)	Switching Kalman Filter	• Tempo: IOI • Dynamics: MIDI velocity	• Low-level: position in the score
Polyhymnia Kim et al. (2013)	3 CRFs modeling the highest and lowest voices (m) and a harmony model (h) for the inner voices (+ rules for dynamics markings and ornaments from statistical analysis)	• Tempo: log IBI ratio (m) • Timing: onset deviations • Articulation: note-wise duration ratio (m, h). • Dynamics: log MIDI velocity ratio (m, h).	• Low-level: pitch (m), duration (m,h), interval to outer voices (h) • High-level: metrical strength (m)
Laminae Okumura et al. (2014)	Performance cases modeled by Gaussian distributions and Tree-based clustering + first order Markov model.	Voice-wise first differences of • Tempo: ave. bpm per beat • Timing: onset deviations (bpm) • Articulation: duration ratio • Dynamics: MIDI velocity	• Low-level: pitch class, octave, dynamics markings • High-level: phrasing, voice (human annotated)
SONY Moulieras and Pachet (2016)	Maximum Entropy model	• Timing: onset deviation. • Dynamics: MIDI velocity	• Low-level: onset position in the bar

Artificial Neural Network-based Approaches Broadly speaking, artificial neural networks (ANNs) can be understood as a family of mathematical functions describing hierarchical non-linear transformations of their inputs. The success of ANNs and deep learning in other areas, including computer vision and natural language processing (Goodfellow et al., 2016) and music information retrieval (Humphrey et al., 2012; Schlüter, 2017), has motivated their use for modeling expressive performance in recent years. A description of systems using ANNs for performance generation is given in Table 9.2. As in the case of probabilistic models, the list of score features for each model is not exhaustive.

Some ANN-based approaches use feed forward neural networks (FFNNs) to predict expressive parameters as a function of the score features (Bresin, 1998; Cancino Chacón and Grachten, 2015; Giraldo and Ramírez, 2016a). These systems tend to compensate for the static nature of FFNNs by including score features that describe some of the musical context of a performed note (e.g. features describing the adjacent rhythmic/melodic context). Other approaches use recurrent neural networks (RNNs), a class of dynamic ANNs, to model temporal dependencies between score features and expressive parameters (Cancino Chacón and Grachten, 2016; Cancino-Chacón et al., 2017d). While early versions of the *Basis Mixer*, an implementation of the Basis Function model, used a simple linear model (Grachten and Widmer, 2012; Krebs and Grachten, 2012), current incarnations (Cancino Chacón and Grachten, 2016) use both FFNNs and RNNs as non-linear function classes, either in the form of deterministic ANNs, or as Gaussian mixture density networks – probabilistic ANNs in which the outputs of the network parameterize the joint probability distribution of a Gaussian Mixture Model.

Neural network models closely follow probabilistic approaches in terms of their expressive parameters. Instead of expecting a human-annotated or heuristically computed decomposition of a polyphonic score into monophonic streams, Cancino Chacón and Grachten (2016) decompose a performance into a series of sequential and non-sequential expressive parameters, which permits to model both temporal trends in dynamics and tempo, and local effects (note-level) in timing, articulation and dynamics deviations. Giraldo and Ramírez (2016a) present an approach for modeling jazz guitar performances which allows for describing not only dynamics and timing, but also ornamentation.

In terms of input features, most ANN models again tend to rely on a combination of low-level hand-crafted features describing local aspects describing individual notes, and some higher-level features relating to structural properties of the music. On the other hand, some researchers have tried to use ANNs to automatically *learn* features from low-level representations of the score. Grachten and Krebs (2014) and van Herwaarden et al. (2014) use Restricted Boltzmann Machines (RBMs), a probabilistic class of neural networks, to learn features from note-centered piano rolls in an unsupervised fashion.

9.3.2 Expressive Interactive Systems: Models for Technology-mediated Performance

A second major trend that can be observed in recent years is a growing interest in developing human–computer interaction systems that generate expressive music performances. Rowe (1992) proposed a terminology for categorizing interactive music systems in three dimensions: *score-driven* vs. *performance-driven*, referring to whether the system follows a musical score or responds to a human performance; *instrument paradigm* vs. *player paradigm*, if the system is meant for solo or ensemble performances; and *transformative* vs. *generative* vs. *sequenced*, describing how the system renders the music. The focus of the present survey is on expressive score-driven systems; performance-driven approaches such as interactive improvisation systems

Table 9.2: Neural Network Based Models

System	Computational Model	Expressive Parameters	Score Features
Bresin Bresin (1998)	FFNNs	• Tempo: IOI • Articulation: performed duration • Dynamics: change in loudness	• Low-level: pitch, duration, melodic interval • High-level: encodings of conditions for KTH rules, like leap articulation, melodic charge, articulation repetition
Unsupervised RBM van Herwaarden et al. (2014); Grachten and Krebs (2014)	RBM (features) + Linear Models	• Dynamics: MIDI velocity	• Low-level: note-centered piano-roll, MIDI-velocity history
Giraldo & Ramírez Giraldo and Ramírez (2016a)	• Ornamentation (classification): FFNNs, decision trees, SVMs, k-NN • Timing, Articulation and Dynamics (regression): FFNNs, regression trees, SVMs, k-NN	• Timing: onset deviation • Articulation: duration ratio • Dynamics: energy ratio • Ornamentation	• Low-level: pitch, duration, position in bar • High-level: Narmour's IR features, key, metrical position, phrase position
Basis Mixer Grachten and Widmer (2012); Cancino Chacón and Grachten (2016)	• Onset-wise model: RNNs • Note-wise: models FFNNs Models can be either deterministic NNs or probabilistic GMDNs	• Tempo: log-IBI (onsetwise) • Timing: Onset deviations (notewise) • Articulation log-duration (notewise) • Dynamics: MIDI velocity trend (onsetwise) and deviations (onsetwise)	Encoding of score aspects through basis functions. • Low-level: pitch, duration, dynamics and articulation markings, position in bar • High-level: tonal tension, harmonic analysis

are beyond the scope of this chapter. A more thorough review of interactive systems is provided by Chew and McPherson (2017).

Conductor Systems

Conductor systems allow the user to shape a solo performance in real-time, and in Rowe's taxonomy would classify as score-driven, instrument paradigm, transformative systems. Such models divide the rendering of an expressive performance into three parallel subtasks: capturing the input from the user, mapping such input to expressive parameters, and providing feedback to the user in real time. Table 9.3 shows several feedback and conductor models. For a more thorough review of feedback models we refer the reader to Fabiani et al. (2013).

Common ways for a user to control certain aspects of a performance are either via high-level semantic descriptors that describe the intended expressive character – often selected from some 2D space related to Russell (1980)'s valence–arousal plane (Friberg, 2006; Canazza et al., 2015); or via physical gestures, measured either through motion capture (Fabiani, 2011) or by using physical interfaces (Chew et al., 2005; Dixon et al., 2005; Baba et al., 2010). Some systems even attempt to provide a realistic simulation of conducting an orchestra (Fabiani, 2011; Baba et al., 2010).

Regarding the mapping of expressive intentions to performance parameters, some systems give the performer direct control of expressive parameters (e.g. tempo and MIDI velocity) via their input (Chew et al., 2005; Dixon et al., 2005). This allows for analyzing the way humans perform music (Chew et al., 2005, 2006). On the other hand, most systems use rule-based models, like the KTH model, to map the user input to expressive parameters (Friberg, 2006; Baba et al., 2010; Fabiani, 2011; Canazza et al., 2015).

Accompaniment Systems

Accompaniment systems are score-driven, player paradigm systems, according to Rowe's taxonomy. In order to successfully perform together with a human, accompaniment systems must solve three tasks: detecting the solo part, matching the detected input to the score, and generating an expressive accompaniment part (Dannenberg, 1984). The first tasks refers to the ability of the system to capture a human performance in real time (either from a microphone or a MIDI instrument) and identify the performed notes, while the second refers to matching these performed notes to notes in the score (also in the presence of errors). The third task involves generating an expressive accompaniment that adapts to the performance of the soloist. The first two tasks are commonly referred to as real-time *score following*. In this review we focus mostly on accompaniment systems for notated Western classical music. For perspectives on accompaniment systems for popular music, we refer the reader to Dannenberg et al. (2014).

Perhaps the most well-developed accompaniment systems are *Antescofo* (Cont, 2008; Cont et al., 2012) and *Music Plus One* (Raphael, 2001a,b, 2010). Antescofo is not only a polyphonic accompaniment system, but a synchronous programming language (i.e. a computer language optimized for real-time reactive systems) for electro-acoustical musical composition. Both systems solve the score following problem using dynamic probabilistic graphical models such as variants of HMMs and DBNs. *Eurydice* (Nakamura et al., 2013, 2014a, 2015a,b) is a robust accompaniment system for polyphonic music that allows for skips, repetitions and ornaments using hidden semi-Markov models.

In spite of the great progress in automatic accompaniment systems, Xia (2016) points out that

Table 9.3: Feedback and conductor systems

System	Computational Model	User Input	Feedback	Expressiveness controlled by the user
pDM Friberg (2006)	Rule based: • KTH model: user controlled weighting of the rules	Semantic descriptors (Russell's space)	Audio (MIDI)	• Tempo • Dynamics • Timing • Articulation
Home Conducting Friberg (2005)		Gestures	Audio (MIDI) and Visual (emotion)	
PerMORfer Fabiani (2011)		Gestures/ semantic descriptors (Russell's space)	Audio (modified recordings)	
Air worm Dixon et al. (2005)	Direct control	MIDI theremin	Audio (MIDI) / Visual (performance worm)	• Tempo • Dynamics
ESP-Chew Chew et al. (2005, 2006)	Direct control	Car controller interface (steering wheel, gas and brake pedals)	Audio / Visual (road simulation)	• Tempo (car control) • Dynamics (car acceleration)
Virtual Philharmony Baba et al. (2010)	Rule based: • Linear models defining tempo adjusting heuristics	Physical Interface	Audio	• Tempo
Caro 2.0 Canazza et al. (2015)	Rule based: • Naturalizer: rules controlling base performance • Expressivizer: user controlled expressive deviations using linear models	Semantic Descriptors (Russell's space)	Audio	• Tempo • Dynamics • Articulation

most of the work on accompaniment systems has focused on solving the score following problem, while overlooking the generation of expressivity in the accompaniment part, or mostly focusing on expressive timing. However, in recent years there has been a growing interest in expressive accompaniment systems. Specifically, Xia and Dannenberg (2015) and Xia et al. (2015) show how to use linear dynamical systems trained via spectral learning, to predict expressive dynamics and timing of the next score events. The *ACCompanion* (Cancino-Chacón et al., 2017a) is a system that combines an HMM-based monophonic score follower with a variant of the Basis Mixer to predict expressive timing, dynamics, and articulation for the accompaniment.

Another interesting recent development in accompaniment systems is embodied human–computer interactions through *humanoid robots* (Hoffman and Weinberg, 2011; Lim et al., 2012; Solis and Takanishi, 2013; Xia, 2016). These robots could be used for studying the way humans interact with each other.

9.3.3 Use of Cognitively Plausible Features and Models

A third clearly recognizable trend in performance modeling has to do with using features and models inspired by music psychology and cognition. While in early work (e.g., Widmer, 2003) the focus was on features rooted in music theory, such as scale degrees, melodic intervals and metrical positions, recent years have seen an increased interest in developing descriptors that capture some aspects of the way humans – both listeners and performers – hear music. Wiggins et al. (2010) suggest that music theory is a kind of folk psychology, and thus, might benefit from being more explicitly informed by music cognition. The music cognition literature supports the hypothesis that much of the way we perform music is informed by the way we perceive music (Farbood, 2012; Goodchild et al., 2016).

Cognitively Inspired Features

From a computational modeling perspective, perhaps the most straightforward approach towards cognitively plausible models is to use features related to aspects of cognition. An important aspect of music cognition is the *expectation* of musical events. One of the most commonly used frameworks of music expectation in computational models of expression is Narmour's *Implication–Realization (IR) model* (Narmour, 1990). The IR model is a music-centered cognitive framework based on Gestalt theory that has emerged from Schenkerian analysis. It defines a number of patterns of listeners' ongoing expectations regarding the continuation of a melody, and how these expectations can be realized to different degrees by the actual continuation. Methods that include features based on IR include YQX (Flossmann et al., 2013), Giraldo and Ramírez (2016b)'s approach to studying expression in Jazz guitar, and Marchini et al. (2014)'s approach for string quartets. More recently, there has been an interest to use information theoretic features computed using the *IDyOM model* (Pearce, 2005), a probabilistic model of statistical learning whose expectations have been shown to match human listeners'. Gingras et al. (2016) use entropy and information content as features to study expressive timing and perceived tension. This work supports Kendall and Carterette (1990)'s hypothesis regarding the communication between the composer, the performer and the listener by linking expectation features, defined by the composer, to expressive timing, controlled by the performer, which is linked to perceived tension by the listener. Cancino-Chacón et al. (2017c) explore the use of these information-theoretic features for actually predicting expressive tempo and dynamics of polyphonic piano music.

Other related cognitive aspects that influence the way humans perform music are the perception

of tonality and tonal tension (Farbood, 2012; Chew, 2016). Several systems incorporate features relating to the tonal hierarchies defined by Krumhansl and Kessler's profiles (Krumhansl, 1990), including YQX (Flossmann et al., 2013), (Giraldo and Ramírez, 2016b) and the Basis Function models (Cancino Chacón and Grachten, 2016; Cancino-Chacón et al., 2017d; Cancino-Chacón and Grachten, 2018), which also include tonal tension features by Herremans and Chew (2016) to predict expressive tempo, timing, dynamics and articulation.

Cognitively Inspired Models

On the other hand, some researchers incorporate aspects of cognition as part of the design of the computational model itself. Recent versions of the KTH model includes some rules that refer to musical *accents* (Bisesi et al., 2011), local events that attract a listener's attention through changes in timing, dynamics, articulation or pedaling; and musical tension rules (Friberg et al., 2006). The approach presented by Gu and Raphael (2012) decomposes expressive timing into discrete "behaviors": constant time, slowing down, speeding up and accent, which, as the authors argue, are more similar to the way human performers conceptualize expressive performance actions. *Polyhymnia* (Kim et al., 2013) uses 3 Conditional Random Fields (CRFs) to independently model the highest, lowest and internal voices. This decomposition allows the model to define the expressive parameters for the internal voices in terms of the outermost voices, following the hypothesis that listeners perceive the expressivity of the uppermost and lowermost voices more clearly than that of the inner voices (Huron and Fantini, 1989).

9.3.4 New Datasets

Data-driven modeling requires data – in the present case, corpora of music performances from which aspects of expressive performance can be readily extracted. This is a non-trivial problem, particularly for notated music, since performances not only have to be recorded (as audio or MIDI files), but they also have to be aligned to the corresponding score, so that we obtain a mapping between elements in the performance (temporal position in the case of audio recordings, or MIDI pitch, onset and offset times) and elements in the score (score position, or an explicit mapping between a performed MIDI note and a note in the score). This is required in order to be able to calculate, e.g., expressive timing as the deviation of played on- and offsets from the corresponding time points implied by the score.

Table 9.4 presents some of the datasets used for modeling expressive performances in current research. Note that this list is not exhaustive; it is intended to give representative examples of the kinds of existing datasets. Performance datasets can be characterized along various dimensions, which are also shown in Table 9.4:

1. **Instrumentation and Solo/Ensemble Setting**. Performance datasets can include a variety of instruments, ranging from solo to ensemble performances. By far the most studied instrument in computational modeling is the piano, partially due to the existence of computer-controlled instruments such as the Bösendorfer SE/CEUS or the Yamaha Disklavier. However, recently there is also an increased interest in modeling ensembles (Marchini et al., 2014; Liem and Hanjalic, 2015; Grachten et al., 2017). For datasets relating to ensemble performances, an important distinction is between those which only reflect collective characteristics of the performance (as might be the case with datasets containing audio recordings where, e.g., timing and loudness of individual instruments are hard or even impossible to disentangle), and datasets where note-precise data is captured

Table 9.4: Datasets of expressive performances.

Dataset	Performer	Pieces	Genre & Epoch	Multiple Performances		Source	Score Alignment
				different performer	same performer		
PIANO							
Repp Repp (1996)	Advanced students	4	Classical: 1830–1920	yes	yes	computer-controlled piano	note-wise
Vienna 4x22 Goebl (1999)	Advanced students & professionals	4	Classical: 1780–1840	yes	no	computer-controlled piano	note-wise
Batik/Mozart Widmer and Tobudic (2002)	Professional	30+	Classical: 1750–1800	no	no	computer-controlled piano	note-wise
Magaloff/Chopin Flossmann et al. (2010)	World-renowned	150+	Classical: 1800–1850	no	no	computer-controlled piano	note-wise
Zeilinger/Beethoven Cancino-Chacón et al. (2017d)	Professional	15+	Classical: 1790–1830	no	no	computer-controlled piano	note-wise
Mazurka Sapp (2007)	World-renowned	45+	Classical: 1800–1850	yes	some	audio recordings	onset-wise
CrestMuse PEDB Hashida et al. (2008, 2017)	World-renowned & professionals	40+	Classical: 1700–1900	no	yes	audio recordings and computer-controlled piano	note-wise
LeadSheet Moulieras and Pachet (2016)	Professional	170+	Popular: 1950–2000	no	yes	computer-controlled piano	note-wise
e-Piano Competition Simon et al. (2017)	Professional	900+	Classical: 1700–2000	yes	no	computer-controlled piano	none
Xia Xia and Dannenberg (2015)	Advanced Students (duets)	3	Popular: 1800–1990	yes	yes	computer-controlled piano	note-wise
OTHER							
RCO/Symphonic Grachten et al. (2017)	World-renowned (orchestra)	20+	Classical: 1800–1900	yes	no	audio recordings	onset-wise
Marchini Marchini et al. (2014)	Professional (string quartet)	1	Classical: 1790–1800	no	yes	audio recordings	onset-wise

for each performer in the ensemble (as is the case with the Xia dataset described in (Xia and Dannenberg, 2015)).

2. **Performer(s)**. Research on music performance has studied a wide range of musical skill levels, from novices and amateurs to advanced music students (i.e. enrolled in advanced undergraduate and post-graduate music programs), professionals and world-renowned performers. (Whether the performances by 'world-renowned' performers are in any way better than those of 'professional' performers, or who qualifies as a famous artist, are, of course, subjective matters.). Again, "performer" might not be singular, as some datasets relate to ensemble performances (cf. the Xia, Marchini and RCO/Symphonic datasets in Table 9.4).

3. **Genre and Epoch** refer to the kind of music contained in the database and the period in which the music was composed. Most of the work on expressive performance modeling has focused on 19th century Western classical music. In Table 9.4, "Classical" denotes Western classical music and "Popular" denotes music genres such as jazz, folk, rock and pop.

4. **Multiple Performances**. Different musicians perform the same pieces in different ways, and it is highly unlikely that the same performer would generate exactly the same performance more than once. Datasets that include multiple performances of the same piece by different performers allow modeling commonalities and systematic differences among performers, while multiple performances of a piece by the same performer could bring insights into the different aspects that contribute to specific realizations of expressive performance actions.

5. **Source** refers to whether the performances are taken from audio recordings or played on a computer-controlled instrument. Another related issue is whether the performances are recorded in front of a live audience, in a recording studio, or in a research lab. Such differences may have an influence on expressive parameters (Moelants et al., 2012).

6. **Alignment** refers to whether there is a mapping between elements in the performance and the score. (Producing such mappings is generally a very tedious task.) Alignments can be *note-wise*, i.e., individual performed notes are matched to their corresponding symbolic representations in the score; or *onset-wise*, where there is just a mapping between temporal position in the performance and score position.

In spite of the apparent richness and variety of data, it is important to raise awareness to some issues, like the fact that it is unlikely that the same performance would happen in two different kinds of rooms with different audiences (Di Carlo and Rodà, 2014). Furthermore, in the case of computer-controlled instruments, the mapping from MIDI velocities to loudness and timbre is dependent on the instrument.

But perhaps one of the most pressing issues is the availability of the datasets. Part of the impressive progress in other Artificial Intelligence domains is due to the availability of large standard datasets, which allow for comparing different approaches. In our case, however, only a few of the performance datasets are publicly available, often due to rights issues. (Of the datasets reported in Table 9.4, only CrestMuse PEDB, Xia, Vienna 4x22, Mazurka and the e-Piano competition datasets are publicly available). A noteworthy effort towards the compilation of large and varied performance datasets is being made by the CrestMuse group in Japan (Hashida et al., 2017), who not only provide a second edition of the PEDB database, but also have provided some tools for aligning MIDI performances to scores (Nakamura et al., 2017). A more in-depth review of methods for extracting information from performances can be found in (Goebl et al., 2008) and (Goebl and Widmer, 2009).

In particular for score-based music it is of capital importance to have central datasets that combine score information, structural annotation, performance data and performance annotation. Crowd-sourcing platforms for creating and maintaining such databases are an avenue that should definitely be pursued.

9.3.5 Computational Models as Tools for Music Education

A final recent trend we would like to mention here is the increased interest in exploring computational models of expressive performance for educational purposes. Juslin (2003) already pointed out that insights learned by developing such models can help understand and appreciate the way musicians perform music expressively. Furthermore, initiatives like the RenCon competition have stressed from the beginning the importance of using computational models for educational purposes, stating in a tongue-in-cheek manner that RenCon's long-term goal is to have a human performer educated using a computational system win the first prize at the International Chopin Piano Competition by 2100 (Hiraga et al., 2002).

A possible use of computational models as tools for education is to analyze performance strategies from visualizations of expressive parameters (Langner and Goebl, 2003; Grachten et al., 2009; Chew, 2012, 2016) or comparing characteristics of a performance (Sapp, 2007, 2008; Liem and Hanjalic, 2015; Grachten et al., 2017). By highlighting similarities and variations in expressive patterns and qualities in performances and relating these to aspects of the written score, this kind of analyses might be interesting not only to music students, buy also to general audiences, stimulating listeners' engagement with, and understanding of, music. All of this could be built into *active music listening interfaces* (Goto, 2007), such as the integrated prototype of the PHENICX project[3] (Liem et al., 2015).

Computer accompaniment systems can help musicians to practice. First concrete examples are commercial applications such as Smartmusic[4], which commercializes Roger Dannenberg's research, Cadenza[5], based on work by Chris Raphael and Antescofo (Cont, 2008), which has been developed into commercial applications for providing adaptable backing tracks for musicians and music students[6]. Conductor and feedback systems can be also be used for educational purposes, either as a simulation of orchestra conducting for conducting students (Peng and Gerhard, 2009; Baba et al., 2010), or as interactive experiences for helping to introduce general audiences to classical music (Sarasúa et al., 2016).

Another dimension is the technical and mechanical aspects of instrument playing and practicing. Here, for example, algorithms that can determine the difficulty of a piece (Sébastien et al., 2012; Nakamura et al., 2014b) or propose appropriate fingering strategies (Al Kasimi et al., 2007; Balliauw et al., 2015; Nakamura et al., 2014b) would be useful. Furthermore, computational models might help determine a performer's skill level (Grindlay and Helmbold, 2006; Caramiaux et al., 2017). Musical e-learning platforms such as Yousician[7] and Music Prodigy[8] (and many more, as this is a rapidly growing business segment for start-ups) might benefit from models of performance to provide a more engaging experience, as well as to develop better musicianship.

[3] http://phenicx.com
[4] https://www.smartmusic.com.
[5] http://www.sonacadenza.com
[6] https://www.antescofo.com.
[7] https://yousician.com.
[8] https://www.musicprodigy.com.

9.4 A Critical Discussion of Parameter Selection and Model Evaluation

The following section presents a discussion of how certain choices in the score features, expressive parameters and models affect what a computational performance model can describe. We focus on three main aspects, namely, the effects of the choice of expressive targets (Section 9.4.1), the level at which a system models expressive performance, based on Palmer's categories described in 9.2.2 above (Section 9.4.2), and on the way models are evaluated (Section 9.4.3).

9.4.1 Encoding Expressive Dimensions and Parameters

As explained in Section 9.2.3 above, expressive parameters are numerical descriptors that capture certain aspects of a performance. As already discussed by De Poli (2004) (and this remains true today), there seems to be no consensus on the best way of describing a music performance. Instead, each formulation uses variants of these parameters, which has some consequences on the kinds of performances or performance aspects that can be modeled.

The most commonly modeled performance aspects (for the piano) are expressive tempo/timing, dynamics and articulation. To keep the discussion manageable, we will also restrict ourselves to these parameters here, leaving out other dimensions such as timbral parameters, vibrato, or intonation. A piano performance can be represented in the most simplistic way by the three MIDI parameters note onset, offset, and key velocity. Other instruments might involve other parameters such as bow velocity for string instruments (Marchini et al., 2014). Furthermore, in some instruments, like winds and strings, there might be a discussion whether to model perceptual or physical onsets (Vos and Rasch, 1981), or indeed whether the notion of a well-defined, exact onset time is meaningful.

Tempo and Timing

Expressive tempo and timing ultimately relate to the 'temporal position' of musical events. Broadly speaking, *tempo* refers to the approximate rate at which musical events happen. This may refer to the *global tempo* of a performance (which is often roughly prescribed in the score by the metronome number), or to *local tempo*, which is the rate of events within a smaller time window and can be regarded as local deviations from the global tempo. *Expressive timing*, finally, refers to deviations of the individual events from the local tempo. Setting these three notions apart is of crucial importance in quantitative modeling of performance, computational or otherwise.

There is support from the music psychology literature that timing patterns are tempo-dependent (Desain and Honing, 1994; Repp et al., 2002; Honing, 2005; Coorevits et al., 2015). Although there is no clear-cut definition of where local tempo variations end and expressive timing starts, the distinction between local tempo and timing was shown to be perceptually relevant in a study by Dixon et al. (2006) where listeners rated beat trains played along with expressive performances, and were shown to prefer slightly smoothed beat trains over beat trains that were exactly aligned to the note onsets. This reinforces the idea that note level irregularities should be not be regarded as as micro-fluctuations of local tempo, but rather as *deviations* from local tempo. A similar result was presented by Gu and Raphael (2012). Honing (2005, 2006) provides valuable insight into the limits of expressive timing by observing that very strong deviations from a steady beat may interfere with the rhythm that is perceived by the listener. Assuming that a goal of the performer is to make the listener accurately recognize the rhythmic categories

of the score being played, this constrains the freedom of expressive timing. Honing (2006) then uses a model of human rhythm perception to infer limits on expressive timing for phrase endings based on their rhythmic patterns.

Several computational models explicitly separate tempo and timing. Recent versions of the KTH model (Friberg et al., 2006, see Table 1) have rules dealing with tempo (e.g. phrasing rules) and timing (e.g. melodic sync, micro-level timing). In *Laminae* (Okumura et al., 2014), tempo is represented by the average BPM per beat, while timing is defined as the onset deviations relative to the beat. *Polyhymnia* (Kim et al., 2011) decomposes tempo into two expressive parameters, calculating tempo curves for the highest and lowest melodic lines. *YQX* (Flossmann et al., 2013) represents tempo as the lower frequency components of the log-IOI ratio series, and timing as the residual high frequency components. In a similar fashion, the most recent version of the *Basis Mixer* (Cancino Chacón and Grachten, 2016) computes expressive tempo from the smoothed log-IOI series, where the estimated IOIs come from a smoothed (spline) interpolation of the performed onsets, and timing as the deviations from these estimated IOIs. There are some practical issues with the use of smooth tempo targets, such as the problem of phrase boundaries, where tempo changes are not necessarily smooth. A solution involving adaptive smoothing (Dixon et al., 2006) – splines with manual knot placement at phrase boundaries – would require human annotation of the phrase structure. Dannenberg and Mohan (2011) describe an interesting dynamic programming optimization algorithm to find the best spline fit allowing a finite number of knots without manual annotations. Other approaches involve local linear approximations of the tempo (Xia, 2016) or multiple hierarchical decompositions (Widmer and Tobudic, 2002).

Another issue related to the modeling of tempo and timing is *scaling* of the expressive parameters, which determines whether we model relative tempo changes, or the actual tempo itself. Chew and Callender (2013) argue in favor of using log-tempo for analysis of performance strategies. Flossmann et al. (2013), Kim et al. (2013) and Grachten and Cancino-Chacón (2017) use logarithmic tempo parameters, while most works focus on linear parameters (Grindlay and Helmbold, 2006; Teramura et al., 2008; Gu and Raphael, 2012; Okumura et al., 2014; Gingras et al., 2016; Cancino-Chacón et al., 2017c; Peperkamp et al., 2017).

Some choose to focus on modeling the dynamic *change* in the parameters instead of the parameters themselves, by calculating *differences*. Gingras et al. (2016) model both IOIs and their first differences – also for a technical reason, since the IOI series is not stationary, and thus not suitable for linear time-series analysis. Okumura et al. (2014) focus on the changes in expressive tempo, by explicitly modeling the conditional probability distribution of the current expressive tempo given its previous difference, using Gaussian distributions. Grindlay and Helmbold (2006) jointly model expressive tempo and its first differences, which leads to more coherent predictions.

Articulation

Articulation, in the case of the piano, refers to the ratio between the performed duration of a note and its notated value and therefore also describes the amount of overlap between consecutive notes. Common articulation strategies include ***staccato*** (shortening compared to notated duration) and ***legato*** (smooth connection to following note). While most generative models deal with expressive tempo/timing, not all of them model articulation. As with tempo, there are several variants of quantitatively describing articulation, including the use of linear (Flossmann et al., 2013) or logarithmic scaling of the parameters (Kim et al., 2011; Cancino Chacón and Grachten, 2016).

To the best of our knowledge, no data-driven generative system has attempted to model *pedaling*, a subtle art that has complex consequences for note durations, but also for the overall sound of a passage. The effect of pedaling on articulation may still be modeled implicitly, by distinguishing between the events of a piano key release and the actual ending of the associated sound (when the sustain pedal is released), as is done in the Basis Function models, for example.

Expressive Dynamics

To simply relate performed dynamics to loudness would miss a number of important aspects of expressive performance. As discussed by Elowsson and Friberg (2017), there is a difference between mere loudness and perceived dynamics. For example, it has been noted that the timbral characteristics of instruments (and therefore, their spectra) change with the performed intensity. Liebman et al. (2012) choose not to focus on loudness since analysis of loudness might not be entirely reliable.

Most approaches for the piano use MIDI velocity as a proxy for loudness. However, it must be noted that the mapping of MIDI velocities to performed dynamics and perceived or measured loudness in piano is not standardized in any way – it may be non-linear, and change from instrument to instrument. Some systems simply use MIDI velocity as an expressive target for each note, while others – particularly those for polyphonic music – decompose the MIDI velocity into several parameters. Early versions of the Basis Function model (Grachten and Widmer, 2012; Cancino Chacón and Grachten, 2015), as well as the unsupervised approach by van Herwaarden et al. (2014) and the NAIST model (Teramura et al., 2008), are non-sequential models and thus predict MIDI velocity for each score note. Sequential models such as ESP (Grindlay and Helmbold, 2006), Laminae (Okumura et al., 2011) and Polyhymnia (Kim et al., 2011) decompose a piece of music into several melodic lines, either automatically (Polyhymnia) or manually (ESP, Laminae), and predict the MIDI velocity for each voice independently. The latest version of the Basis Function models decomposes a performance into a dynamic trend, either the average or the maximal MIDI velocity at each score position (Cancino Chacón and Grachten, 2016; Cancino-Chacón et al., 2017c), and a local parameter describing the deviations from the trend for each score note. The rationale for this decomposition is that it allows for modeling the temporal evolution of expressive dynamics, something that cannot easily be done in polyphonic music when dynamics is represented as an attribute of individual notes.

In the case of *audio*, the problem of choosing a metric for expressive dynamics is more complicated due to the large number of measures of loudness. A common trend is to use loudness measures that take into account human perception, such as the EBU R 128 measure defined for regulation of loudness in the broadcasting industry (Grachten et al., 2017), and smoothed loudness curves in sones (Kosta et al., 2016).

Joint Modeling of Parameters

Musicians' expressive manipulations of tempo, timing, dynamics, and articulation have been studied from a cognitive perspective, both individually and in combination, to determine how they shape listeners' perceptions of performed music. A number of studies have sought to identify interactions between pairs of expressive parameters like timing and dynamics (Tekman, 2002; Boltz, 2011), and timing and tempo (Desain and Honing, 1994; Repp et al., 2002; Coorevits et al., 2015, 2017). While the music psychology literature provides some indication of how listeners expect pairs of expressive parameters to relate in certain (simplistic) contexts, it remains unclear

whether these relationships are upheld during normal music performance, when the underlying piece is complex and many expressive parameters must be manipulated in parallel.

The influential model of expressive tempo and dynamics by Todd (1992) states that both aspects are linearly coupled by default (unless the musical context demands a decoupling), and suggests that this coupling may be especially tight for romantic piano music. The model predicts arc-like dynamics and tempo shapes to express phrase structure. Grindlay and Helmbold (2006)'s HHMM-based ESP system allows for the joint modeling of expressive parameters, however the focus in their work is strongly on local tempo. No quantitative results are given for the modeling of tempo in combination with dynamics and articulation. The KTH model (Friberg et al., 2006) includes rules that prescribe the joint variation of multiple parameters, such as a *phrasing* rule that accounts for arc-like shapes in dynamics and tempo, similar to those in Todd (1992). Several other authors combine separate models for each expressive parameter, and do not consider interactions (Teramura et al., 2008; Widmer et al., 2009), or consider only a single expressive parameter (Kosta et al., 2016; Peperkamp et al., 2017). Recent versions of the Basis Function models (Cancino Chacón and Grachten, 2016) allow for joint estimation of parameters using Gaussian mixture density networks (GMNs); parameters defined for individual notes and parameters defined only per score time point are modeled in separate sets. Xia and Dannenberg (2015) and Xia et al. (2015) jointly model expressive dynamics and tempo using linear dynamical systems, with the underlying assumption that the joint distribution of the parameters is Gaussian. The approach presented by Moulieras and Pachet (2016) models dynamics and timing jointly with a joint probability distribution approximated using a maximum entropy approach. Since this approach is not Gaussian, the form of the distribution depends on the training data.

To the best of our knowledge, there has not been an extensive computational study analyzing whether the joint estimation of parameters improves the generative quality of predictive models. Furthermore, in some cases performers will manipulate two parameters in different ways during the course of a single piece to achieve different expressive goals (e.g. slowing down while simultaneously getting softer, then elsewhere slowing down while getting louder). Whether the consistent use of particular parameter relationships relates to the aesthetic quality of a performance, increases its predictability, or makes the communication of expression more successful likewise requires further study.

9.4.2 Relation to Palmer's Categories

Interpretation

As stated in Section 9.2.2, expressive performance of notated music can be seen as a communication process in which information flows from the composer to the listener through the performer (Kendall and Carterette, 1990). In this case, the role of the performer involves semantically and affectively *interpreting* the score. Gingras et al. (2016) provide evidence supporting this relationship by linking information-theoretic features (related to the role the composer) to expressive timing (performer), which is a good predictor of perceived tension (listener).

An important aspect of the interpretation of a score is to highlight structural content. A common approach taken by many systems is to rely on input features describing *group boundaries* and *phrase structure*. Friberg et al. (2006) and Grindlay and Helmbold (2006) use features related to phrase structure, which is assumed to be manually annotated in the score. Giraldo and Ramírez (2016a,b) use LBDM, an automatic segmentation algorithm based on Gestalt theory (Cambouropoulos, 1997).

Another important aspect in polyphonic Western music is the hierarchical relations and interactions between different *voices*, which in most cases involves distinguishing the main (or most salient) melody. Several models require the melody to be annotated (Grindlay and Helmbold, 2006; Cancino-Chacón et al., 2017c; Okumura et al., 2014). Other models simply assume that the main melody is composed of the highest notes (Teramura et al., 2008; Flossmann et al., 2013).

Another marker of music structure are the patterns of *tension* and *relaxation* in music, linked to several aspects of *expectedness*. Farbood (2012) showed a relationship between expressive timing and perceived tension. Grachten and Widmer (2012) use Narmour's (1990) Implication–Realization model to link expressive dynamics to melodic expectation, but observe no substantial improvement over simpler models that use only pitch and dynamics annotations as predictors. Chew (2016) introduces the idea of tipping points, i.e. extreme cases of pulse elasticity, and their relation to tonality, in particular harmonic tension. The KTH model includes features describing harmonic tension (Friberg et al., 2006). Gingras et al. (2016) show relationship of expressive timing and perceived tension. Recent versions of the Basis Function models (Cancino Chacón and Grachten, 2016) include harmonic tension features computed using the methods proposed by Herremans and Chew (2016).

Beyond the identification of structural aspects, another important aspect of interpretation is to highlight particular *emotional content* of the music. Juslin (2003) points out that "*[a] function of performance expression might be to render the performance with a particular emotional expression*". Research in music and emotion is a very active field (see Juslin and Sloboda (2011) for an overview), which includes studying the relationship between intended emotion and performance gestures and strategies (Juslin, 2001; Gabrielsson and Lindström, 2010; Bresin and Friberg, 2011). Eerola et al. (2013) study the contribution of expressive dimensions such as tempo, dynamics, articulation, register and timbre to determining emotional expression. Their results suggest that expressive dimensions interact linearly, and their contributions seem to be additive. While some generative models allow the user to control the intended emotion or expressive character (Bresin and Friberg, 2000; Friberg, 2006; Canazza et al., 2015), to the best of our knowledge no autonomous generative model attempts to recognize emotive content of a piece directly from analysis of the score, and render it appropriately.

Planning

While interpretation of a musical score aims at uncovering its semantic and affective content, performance planning refers to how this content, along with more or less specific artistic or expressive intentions of the performer, is turned into specific expressive performance decisions. In this view, most computational models of expressive performance act at this level, since they focus on explicitly (i.e. quantitatively) relating structural aspects of the score to parameters encoding an expressive performance.

An important characteristic of Western classical music is the hierarchical nature of its structure. Repp (1998) points out that *"[t]he performer's (often subconscious) intention seems to 'act out' the music's hierarchical grouping structure and thereby communicate to the listeners"*. It is therefore important to determine how the different hierarchical levels interact with each other and contribute to the overall expression. The relation between the hierarchical structure and expression has been explored in the cognitive literature (Clarke, 1993; Repp, 1998; Toiviainen et al., 2010). Widmer and Tobudic (2002) explore the relationship between hierarchical levels of the phrase structure and expressive tempo, using a multilevel decomposition of the tempo curves corresponding to each level of the phrase structure, and an inductive rule learning method to

model the note-wise performance residuals. Tobudic and Widmer (2006) expand on this work using an instance-based learning method in which the hierarchical phrase structure is represented using first-order logic.

An important design issue relating to the structure–expression relationships is how the choice of score (feature) representation affects the possible performance gestures that can be modeled (i.e. planned). An example of this would be whether the possible patterns of dynamics and timing deviations that a system can describe are 'implicitly' assumed from the encoding of features – as might be the case with systems using features describing metrical strength and metrical hierarchy (Grindlay and Helmbold, 2006; Teramura et al., 2008; Kim et al., 2011; Marchini et al., 2014; Giraldo and Ramírez, 2016a) – or can be inferred directly from human performances using more agnostic features denoting metrical position (Xia et al., 2015; Cancino-Chacón et al., 2017d).

Movement

Humans need to transform the result of the interpretation and planning stages into an actual acoustic rendering of a piece by means of movements of their bodies (i.e., actually playing the instrument). In this regard, we can consider movement and embodiment as necessary conditions for (human) expressive performance. Similar to the concept of embodied cognition (Leman et al., 2017a), neuroscientific accounts refer to the "action–perception loop," a well-trained neural connection between the aim of an action, here the musical sound, and its execution, the necessary body movements at the musical instrument (Novembre and Keller, 2014). Musicians, having practiced over decades, will "hear" or imagine a certain sound and execute the appropriate body movements automatically. Likewise, co-musicians or the audience will perceive a performance through hearing and seeing the performer (Platz and Kopiez, 2012); and even from only hearing the sound, experienced listeners will be able to deduce bodily states and movement characteristics of the performer. Leman et al. (2017b) discuss the role of the hand as a co-articulated organ of the brain's action–perception machinery in expressive performance, music listening and learning.

Body motion is an essential means of non-verbal communication not only to the audience, but also among musicians. Goebl and Palmer (2009) showed in ensemble performances of simple melodies that visual information became more important to stay in synchrony (i.e., musicians' head movements were more synchronized) as auditory cues were reduced. Body movements serve specific roles at certain places in a piece (e.g., at the beginning, after fermatas). Bishop and Goebl (2017b,a) study specific head motion kinematics in ensemble performance used to cue-in a piece without upbeat. They found characteristic patters including acceleration peaks to carry relevant cueing information.

In spite of the progress in music psychology and embodied cognition, few computational approaches take into account aspects of motion while modeling expressive performance. However, the availability of motion capture technology as well as new trends in psychological research might open the field of modeling expressive movement. The KTH model includes performance noise as a white noise component relating to motor delay and uses 1/f noise to simulate noise coming from an internal time-keeper clock (Friberg et al., 2006). Dalla Bella and Palmer (2011) show that finger velocity and acceleration can be used as features to identify individual pianists. Marchini et al. (2013, 2014) study expressive performance in string quartets using a combination of music-only related expressive parameters, as well as bow velocity, a dimension of movement directly related to performed dynamics. Caramiaux et al. (2017) assess whether individuality can be trained, that is whether the differences in performance style are related to development

in skill and can thus be learned. Their results suggest that motion features are better than musical timing features for discriminating performance styles. Furthermore, the results suggest that motion features are better for classification.

9.4.3 Evaluating Computational Performance Models

How the quality or adequacy of computational performance models can be evaluated in a systematic and reliable fashion is a difficult question. First of all, the evaluation will depend on the purpose of the model. A model designed to serve an explanatory purpose should be evaluated according to different criteria than a model for performance generation. In the former case, the simplicity of the model structure may be of prime importance, as well as how easily the model output can be linked to aspects of the input. In the latter, we may be more interested in how convincing the generated performance sounds than how easy it is to understand the decisions of the model.

Furthermore, when we evaluate a model by the quality of its output, an important issue is the ultimately subjective nature of judging the musical quality of an expressive performance. And while we might even be able to formulate principles to which a good performance should adhere, it is entirely conceivable that a performance conforming to all these principles fails to please us, or conversely, that a performance defying these principles is nevertheless captivating.

Bresin and Friberg (2013) formulate several more formal aspects of computational performance models that can be evaluated, including their ability to reproduce/reconstruct (specific) human performances and their capacity to adapt to different expressive intentions/contexts.

Attempts at Quantitative, 'Objective' Evaluation

Most of the work described above relies on quantitative evaluation in terms of predictive capabilities and/or goodness of fit, relative to a given set of human performances. These measures tend to focus on the prediction or reconstruction error – e.g., in the form of the correlation or the mean squared error (MSE) between the performance patterns predicted by a model, and a real human performance –, or on a so-called likelihood function (which gives the probability of observing a given (human) performance, given a particular model). What all these approaches have in common is that they base their evaluation on a comparison between a model's output, and a – usually one specific – performance by a human musician (most often additional performances by the same musician(s) from whom the model was learned). This is problematic for several reasons:

- Comparison to a single 'target' performance is highly arbitrary, given that there are many valid ways to perform a piece. A good fit may at least indicate that a model has the capacity of encoding and describing the specific performances by a specific performer (with, presumably, a specific style). A poor fit does not necessarily mean that the model's predictions are musically bad.
- What is more, there is no guarantee that higher correlation, or lower MSE implies a musically better performance, nor indeed that a performance that sounds more similar to the target. Especially *outliers* (single errors of great magnitude) can influence these measures. Errors may not be equally salient for all data points. Accounting for this would require a model of perceived saliency of musical positions and errors, which is currently out of reach (or has not been tackled yet).

- A more technical point is that we cannot compare performance models that encode an expressive dimension using different parameters (such as modeling expressive tempo using IBI vs. BPM, or using linear vs. logarithmic parameters), because quantitative correlation or error measures must assume a particular encoding. There are currently no canonical definitions of the expressive dimensions.

These kinds of problems have also been faced in other domains, and have been addressed in the computer vision literature with the introduction of the Structural Similarity Index (Wang et al., 2004), a perception-based metric that considers perceptual phenomena like luminance masking, as well as perceived change in structural information. However, to the best of our knowledge, there has not been any attempt to define similar measures for music, or to propose a custom measure for expressive performance.

Bresin and Friberg (2013) suggest to relate these error metrics to more general perceptual models. An example of this would be reporting the error in terms of just noticeable differences (JNDs) in the expressive parameters. Nevertheless, it is worth noticing that JNDs are highly dependent on the musical context.

Qualitative Evaluation via Listening Tests

The obvious alternative to quantitative, correlation-based evaluation is evaluation by listening: playing human and computer-generated performances to human listeners and asking them to rate various qualities, or to simply rank them according to some musical criteria.

An early initiative that attempted to provide a systematic basis for this was *RenCon (Performance Rendering Contest)*,[9] a Japanese initiative that has been organizing a series of contests for computer systems that generate expressive performances (Hiraga et al., 2002, 2003, 2004, 2006; Katayose et al., 2012). At these RenCon Workshops, (piano) performances of different computational models were played to an audience, which then voted for a 'winner' (the most 'musical' performance). It is currently not clear if this initiative is being continued. (Actually, the last RenCon workshop we are aware of dates back to 2013.)

Also, there are a number of issues with audience-based evaluation, which clearly surfaced also in the RenCon Workshops: the appropriate choice of music; the listeners' limited attention span; the difficulties inherent in comparing different kinds of systems (e.g., fully autonomous vs. interactive), or systems that model different performance parameters (e.g., not all models address the articulation dimension, or autonomously decide on the overall tempo to be chosen). Finally, reproducibility of results is an issue in audience-based evaluation. There is no guarantee that repeating a listening test (even with the same audience) will yield the same results, and it is impossible to compare later models to models that have been evaluated earlier.

A particular and very subtle problem is the choice, and communication to the human subjects, of the *rating criteria* that should be applied. This can also be exemplified with a recent *'Turing Test'* described in (Schubert et al., 2017), where piano performances produced by several systems that had won recent RenCon competitions, along with one performance by a real human pianist, were rated by a human listening panel. The subjects were asked to rate the performances (all rendered on the same piano) according to different dimensions. The question that the analysis in (Schubert et al., 2017) then mainly focuses was to what degree the listeners believed that *"[t]he performance was played by a human"*. Without going into the details of the results[10], it is clear

[9] www.renconmusic.org

[10] Briefly: it turned out that on this 'perceived humanness' rating scale, several computational models scored at a level that was statistically indistinguishable from the human pianist, with the linear Basis Function

that such a question may be interpreted differently by different listeners, or indeed depending on what apparent weaknesses are heard in a performance: a performance with extreme timing irregularities might be (erroneously) classified as 'human' because the listener might believe that it was produced by a poor piano student or a child, or that it could not be the output of a computer, because computers would be able to play with perfect regularity. Generally, an inherent problem with qualitative listening evaluations is that one single cue (e.g., 'strange', unusual mistake) can give away a given trial as probably computer-generated, independent of how convincing the rest was.

There is plenty of evidence in the music psychology literature showing that the assessment of the quality of a performance depends not only on the quality of its acoustic rendering, but on a number of other factors. Platz and Kopiez (2012) present a meta-analysis of 15 studies from 1985 to 2011 supporting the hypothesis that audio-visual presentation enhances appreciation of music performance. Their results show that the visual component is an important factor in the communication of meaning. Tsay (2013) present controversial results suggesting that visual information alone might be sufficient when determining the winner of a music competition. Wapnick et al. (2009) suggest that certain non-musical attributes like the perceived attractiveness of a performer or the way they behave on stage affects ratings of high-level piano performances, particularly on short performances. Thompson et al. (2007) study the evolution of listeners' assessments of the quality of a performance over the course of the piece. Their results suggest that even while listeners only need a short time to reach a decision on their judgment, there is a significant difference between the initial and final judgments. Wesolowski et al. (2016) present a more critical view of listeners' judgments by examining the precision.

De Poli et al. (2014) and Schubert et al. (2014a) specifically study how the audience judges entire performances of computational models, by analyzing listeners' scores of several aspects including technical accuracy, emotional content and coherence of the performed style. The listeners were categorized into two different cognitive styles: music systemizers (those who judge a performance in technical and formal terms) and music empathizers (describe a performance in terms of its emotive content). Their results suggest that preference for different performances cannot be attributed to these cognitive styles, but the cognitive style does influence the justification for a rating. Schubert et al. (2014b) suggest that the conceptual difference between music empathizers and music systemizers might not be sufficient to capture significant differences in evaluating music performances.

Despite all these problematic aspects, some way of qualitative, expert- or listener-based evaluation of computational performance models seems indispensable, as the quantitative measures described in the previous section definitely fall short of capturing the *musical* (not to mention the emotional) quality of the results. This is a highly challenging open problem for the performance research community – and an essential one.

9.5 Conclusions

This work has reviewed some recent developments on the study and generation of expressive musical performances through computational models. Perhaps the most notable trends are a strong focus on data-driven methods for analysis and generation, which mirrors the trend in other areas such as natural language processing and computer vision; and increased interest in interactive systems, which allow us to explore musical human–computer interactions.

model (Grachten and Widmer, 2012) achieving the highest 'humanness' ratings (higher than even the human performance).

In their current state, computational models of performance provide support for a number of musically and cognitively plausible hypotheses, such as the existence of certain patterns in performance, the importance of attending to the local context in the score (Marchini et al., 2014; Kosta et al., 2016; Grachten and Cancino-Chacón, 2017), and Kendall and Carterette (1990)'s communication model for the role of composer, performer and listener (Gingras et al., 2016). Nevertheless, most approaches focus on mapping local syntactic structures to performance gestures, but are not able to model the longer hierarchical relationships that might be relevant for a full understanding of the music, and its dramatic structure.

There remains much to be done in order to advance the state of the art, and to improve the utility of such computational models – both as vehicles for exploring this complex art, and as practical tools. We would like to end this chapter by briefly highlighting four aspects (out of many) to consider for further research in the immediate future:

1. **Dataset creation**. The power of data-driven methods comes at the cost of requiring large amounts of data, in this case specifically, performances aligned to their scores. As pointed out by Juslin (2003), this might be an issue preventing the advance in computational models of music expression. As discussed in Section 9.3.4, currently available datasets do not yet reach the size (in terms of amount of data and variety) that has been able to boost other domains such as computer vision. Still, progress is being made, with initiatives like those by the CrestMuse group in Japan (Hashida et al., 2017). We would like to encourage the community at large to focus on developing more datasets, in a joint effort.

2. **Expressive Parameters**. As discussed in Section 9.4.1, there is no consensus regarding the encoding of expressive dimensions. Efforts should be made to investigate the effects of the choice of performance parameters encoding, as well as joint estimation of parameters. An interesting direction would be to search for representations that are cognitively plausible (in terms of human real-time perception and memory).

3. **Models of music understanding and embodiment.** As pointed out by Widmer (2017), it is necessary to develop models and features that better capture the long term semantic and emotive relationships that appear in music. This might require to develop better features, including learned features, as well as reframing the computational tasks in terms of approaches like reinforcement learning. Furthermore, more research efforts into developing computational models that include aspects of embodied music interaction might be required.

4. **Evaluation** Having well-established and valid criteria for evaluating different models, and comparing their performance to that of humans, is essential to making progress. In terms of quantitative measures, more work will be required to conduct research that studies the effects and biases involved in the choice of evaluation metrics. Furthermore, it would be interesting to evaluate computational models of expression as models of cognition, not only focusing on how well they reproduce the observed data, but also if the predictions of the model are cognitively plausible (Honing, 2006). Ideally, quantitative measures should relate to perceptually relevant aspects of performances, as perceived by musical listeners. In terms of qualitative, truly musical evaluation, which we consider indispensable, we need more efforts towards establishing venues for systematic evaluation and comparison, like the RenCon workshop and similar initiatives. And again, studies that give us a better understanding of how humans evaluate performances, would be extremely useful.

While at this stage (and perhaps forever) it is more than uncertain whether computational models of performance will ever successfully beat humans in high-profile competitions, as stated as a goal by the RenCon initiative (Hiraga et al., 2002), there is no doubt that understanding the way humans create and enjoy expressive performances is of great value. It is our hope that

the field of research we have attempted to portray here can contribute to such an understanding, and develop useful musical tools along the way.

Bibliography

Academy of Motion Picture Arts and Sciences (2015). The 19th Academy Awards — 1947. www.oscars.org/oscars/ceremonies/1947. Accessed: 13-04-2018.

Adamson, J. (1990). *Bugs Bunny: Fifty Years and Only One Grey Hare*. Henry Holt, New York.

Agres, K., Cancino, C., Grachten, M., and Lattner, S. (2015). Harmonics co-occurrences bootstrap pitch and tonality perception in music: Evidence from a statistical unsupervised learning model. In *Proceedings of the Annual Meeting of the Cognitive Science Society (CogSci 2015)*, Pasadena, CA, USA.

Al Kasimi, A., Nichols, E., and Raphael, C. (2007). A Simple Algorithm for Automatic Generation of Polyphonic Piano Fingerings. In *Proceedings of the 8th International Society for Music Information Retrieval Conference (ISMIR 2007)*, pages 355–356, Vienna, Austria.

Al-Rfou, R., Alain, G., Almahairi, A., Angermueller, C., Bahdanau, D., Ballas, N., Bastien, F., Bayer, J., Belikov, A., Belopolsky, A., Bengio, Y., Bergeron, A., Bergstra, J., Bisson, V., Bleecher Snyder, J., Bouchard, N., Boulanger-Lewandowski, N., Bouthillier, X., de Brébisson, A., Breuleux, O., Carrier, P.-L., Cho, K., Chorowski, J., Christiano, P., Cooijmans, T., Côté, M.-A., Côté, M., Courville, A., Dauphin, Y. N., Delalleau, O., Demouth, J., Desjardins, G., Dieleman, S., Dinh, L., Ducoffe, M., Dumoulin, V., Ebrahimi Kahou, S., Erhan, D., Fan, Z., Firat, O., Germain, M., Glorot, X., Goodfellow, I., Graham, M., Gulcehre, C., Hamel, P., Harlouchet, I., Heng, J.-P., Hidasi, B., Honari, S., Jain, A., Jean, S., Jia, K., Korobov, M., Kulkarni, V., Lamb, A., Lamblin, P., Larsen, E., Laurent, C., Lee, S., Lefrancois, S., Lemieux, S., Léonard, N., Lin, Z., Livezey, J. A., Lorenz, C., Lowin, J., Ma, Q., Manzagol, P.-A., Mastropietro, O., McGibbon, R. T., Memisevic, R., van Merriënboer, B., Michalski, V., Mirza, M., Orlandi, A., Pal, C., Pascanu, R., Pezeshki, M., Raffel, C., Renshaw, D., Rocklin, M., Romero, A., Roth, M., Sadowski, P., Salvatier, J., Savard, F., Schlüter, J., Schulman, J., Schwartz, G., Serban, I. V., Serdyuk, D., Shabanian, S., Simon, E., Spieckermann, S., Subramanyam, S. R., Sygnowski, J., Tanguay, J., van Tulder, G., Turian, J., Urban, S., Vincent, P., Visin, F., de Vries, H., Warde-Farley, D., Webb, D. J., Willson, M., Xu, K., Xue, L., Yao, L., Zhang, S., and Zhang, Y. (2016). Theano: A Python framework for fast computation of mathematical expressions. *arXiv e-prints*, abs/1605.02688.

Baba, T., Hashida, M., and Katayose, H. (2010). "VirtualPhilharmony": A Conducting System with Heuristics of Conducting an Orchestra. In *Proceedings of the 10th International Conference on New Interfaces for Musical Expression, NIME 2010*, pages 263–270, Sydney, Australia.

Bach, C. P. E. (1948). *Essay on the True Art of Playing Keyboard Instruments*. W. W. Norton & Company. Translated by W. J. Mitchell. Original work published 1787.

Balliauw, M., Herremans, D., Palhazi Cuervo, D., and Sörensen, K. (2015). Generating Fingerings for Polyphonic Piano Music with a Tabu Search Algorithm. In *Proceedings of the 5th International Conference on Mathematics and Computation in Music (MCM 2015)*, pages 149–160, London, UK.

Bantula, H., Giraldo, S., and Ramírez, R. (2016). Jazz Ensemble Expressive Performance Modeling. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR 2016)*, pages 674–680, New York, NY, USA.

Bergeron, V. and Lopes, D. M. (2009). Hearing and Seeing Musical Expression. *Philosophy and Phenomenological Research*, 78(1):1–16.

Binet, A. and Courtier, J. (1896). Recherches graphiques sur la musique. L'année Psychologique (2), 201–222.

Bisesi, E., Parncutt, R., and Friberg, A. (2011). An Accent-based Approach to Performance Rendering: Music Theory Meets Music Psychology. In *Proceedings of the International Symposium on Performance Science 2011 (ISPS 2011)*, pages 27–32, Toronto, Canada.

Bishop, C. (1994). Mixture Density Networks. Technical Report NCRG/94/004, Neural Computing Research Group, Dept. of Computer Science and Applied Mathematics, Aston University, Birmingham, UK.

Bishop, C. M. (1995). *Neural Networks for Pattern Recognition*. Clarendon Press Oxford.

Bishop, C. M. (2006). *Pattern Recognition and Machine Learning*. Springer Verlag.

Bishop, L., Cancino-Chacón, C., and Goebel, W. (2017). Mapping Visual Attention of Duo Musicians During Rehearsal of Temporally-Ambiguous Music. In *Proceedings of the International Symposium on Performance Science (ISPS 2017)*, Reykjavik, Iceland.

Bishop, L., Cancino-Chacón, C. E., and Goebl, W. (2018). Visual Signals between Improvisers Indicate Attention rather than Intentions. In *Proceedings of the 15th International Conference on Music Perception and Cognition (ICMPC15 ESCOM10)*, Graz, Austria.

Bishop, L. and Goebl, W. (2017a). Beating time: How ensemble musicians' cueing gestures communicate beat position and tempo. *Psychology of Music*, 46(1):84–106.

Bishop, L. and Goebl, W. (2017b). Communication for coordination: gesture kinematics and conventionality affect synchronization success in piano duos. *Psychological Research*, pages 1–18.

Bogen, J. (2017). Theory and observation in science. In Zalta, E. N., editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, summer 2017 edition.

Boltz, M. G. (2011). Illusory tempo changes due to musical characteristics. *Music Perception*, 28(4):367–386.

Boyd, S. P. and Vandenberghe, L. (2004). *Convex Optimization*. Cambridge University Press.

Bresin, R. (1998). Artificial neural networks based models for automatic performance of musical scores. *Journal of New Music Research*, 27(3):239–270.

Bresin, R. and Friberg, A. (2000). Emotional Coloring of Computer-Controlled Music Performances. *Computer Music Journal*, 24(4):44–63.

Bresin, R. and Friberg, A. (2011). Emotion rendering in music: Range and characteristic values of seven musical variables. *CORTEX*, 47(9):1068–1081.

Bresin, R. and Friberg, A. (2013). Evaluation of Computer Systems for Expressive Music Performance. In Kirke, A. and Miranda, E. R., editors, *Guide to Computing for Expressive Music Performance*, pages 181–203. Springer-Verlag, London, UK.

Burden, R. L. and Faires, J. D. (2001). *Numerical Analysis*. Brooks/Cole, 7 edition.

Burloiu, G. (2016). An Online Tempo Tracker for Automatic Accompaniment Based on Audio-To-Audio alignment and Beat Tracking. In *Proceedings of the Sound and Music Computing Conference (SMC 2016)*, Hamburg, Germany.

Cambouropoulos, E. (1997). Musical rhythm: A formal model for determining local boundaries, accents and metre in a melodic surface. In Leman, M., editor, *Music, Gestalt and Computing*, pages 277–293. Springer Berlin Heidelberg, Berlin, Heidelberg.

Canazza, S., De Poli, G., and Rodà, A. (2015). CaRo 2.0: An Interactive System for Expressive Music Rendering. *Advances in Human-Computer Interaction*, 2015(3):1–13.

Cancino-Chacón, C., Bonev, M., Durand, A., Grachten, M., Arzt, A., Bishop, L., Goebl, W., and Widmer, G. (2017a). The ACCompanion v0.1: An Expressive Accompaniment System. In *Late Breaking/ Demo, 18th International Society for Music Information Retrieval Conference (ISMIR 2017)*, Suzhou, China.

Cancino-Chacón, C. and Grachten, M. (2018). A Computational Study of the Role of Tonal Tension in Expressive Piano Performance. In *Proceedings of the 15th International Conference on Music Perception and Cognition (ICMPC15 ESCOM10)*, Graz, Austria.

Cancino-Chacón, C., Grachten, M., and Agres, K. (2017b). From Bach to The Beatles: The Simulation of Human Tonal Expectation Using Ecologically-Trained Predictive Models. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR 2017)*, Suzhou, China.

Cancino-Chacón, C., Grachten, M., Goebl, W., and Widmer, G. (2018). Computational Models of Expressive Music Performance: A Comprehensive and Critical Review. *To appear in Frontiers in Digital Humanities*.

Cancino-Chacón, C., Grachten, M., Sears, D. R. W., and Widmer, G. (2017c). What Were You Expecting? Using Expectancy Features to Predict Expressive Performances of Classical Piano Music. In *Proceedings of the 10th International Workshop on Machine Learning and Music (MML 2017)*, Barcelona, Spain.

Cancino Chacón, C., Lattner, S., and Grachten, M. (2014a). Developing Tonal Perception Through Unsupervised Learning. In *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR 2014)*, pages 195–200.

Cancino-Chacón, C. E., Gadermaier, T., Widmer, G., and Grachten, M. (2017d). An Evaluation of Linear and Non-linear Models of Expressive Dynamics in Classical Piano and Symphonic Music. *Machine Learning*, 106(6):887–909.

Cancino Chacón, C. E. and Grachten, M. (2015). An Evaluation of Score Descriptors Combined with Non-linear Models of Expressive Dynamics in Music. In *Proceedings of the 18th International Conference on Discovery Science (DS 2015)*, pages 48–62, Banff, AB, Canada.

Cancino-Chacón, C. E. and Grachten, M. (2016). Rendering Expressive Performances of Musical Pieces Through Sampling From Generative Probabilistic Models. Technical Report OFAI-TR-2014-01, Austrian Research Institute for Artificial Intelligence, Vienna, Austria.

Cancino Chacón, C. E. and Grachten, M. (2016). The Basis Mixer: A Computational Romantic Pianist. In *Late Breaking/ Demo, 17th International Society for Music Information Retrieval Conference (ISMIR 2016)*, New York, NY, USA.

Cancino Chacón, C. E., Grachten, M., and Widmer, G. (2014b). Bayesian Linear Basis Models with Gaussian Priors for Musical Expression. Technical Report OFAI-TR-2014-12, Austrian Research Institute for Artificial Intelligence, Vienna, Austria.

Cancino Chacón, C. E. and Mowlaee, P. (2014). Least Squares Phase Estimation of Mixed Signals. In *15th Annual Conference of the International Speech Communication Association (INTERSPEECH 2014)*, Singapore.

Caramiaux, B., Bevilacqua, F., Palmer, C., and Wanderley, M. (2017). Individuality in Piano Performance Depends on Skill Learning. In *Proceedings of the 4th International Conference on Movement Computing (MOCO'17)*, page 14, London, UK. ACM.

Carreira-Perpiñán, M. Á. (2000). Mode-finding for mixtures of Gaussian distributions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 22(11):1318–1323.

Cheng, E. and Chew, E. (2008). Quantitative Analysis of Phrasing Strategies in Expressive Performance: Computational Methods and Analysis of Performances of Unaccompanied Bach for Solo Violin. *Journal of New Music Research*, 37(4):325–338.

Chew, E. (2000). *Towards a Mathematical Model of Tonality*. PhD thesis, Massachusetts Institute of Technology, Cambridge, MA, USA.

Chew, E. (2012). About Time: Strategies of Performance Revealed in Graphs. *Visions of Research in Music Education*, 20.

Chew, E. (2016). Playing with the Edge: Tipping Points and the Role of Tonality. *Music Perception*, 33(3):344–366.

Chew, E. and Callender, C. (2013). Conceptual and Experiential Representations of Tempo: Effects on Expressive Performance Comparisons. In *Proceedings of the 4th International Conference on Mathematics and Computation in Music (MCM 2013)*, pages 76–87, Montreal, QC, Canada.

Chew, E., François, A., Liu, J., and Yang, A. (2005). ESP: a driving interface for Expression Synthesis. In *Proceedings of the 2005 conference on New Interfaces for Musical Expression, NIME 2005*, pages 224–227, Vancouver, BC, Canada.

Chew, E., Liu, J., and François, A. R. J. (2006). ESP: Roadmaps As Constructed Interpretations and Guides to Expressive Performance. In *Proceedings of the 1st ACM Workshop on Audio and Music Computing Multimedia*, pages 137–145, New York, NY, USA. ACM.

Chew, E. and McPherson, A. (2017). PERFORMING MUSIC: Humans, Computers and Electronics. In Ashley, R. and Timmers, R., editors, *The Routledge Companion to Music Cognition*, pages 301–312. Routledge, NY, NY.

Chopin, F. (1915a). Nocturne Op. 9 No. 1. In Joseffy, R., editor, *Complete Works for the Piano, Vol.4: Nocturnes*, pages 5–7. G. Schirmer, Inc.

Chopin, F. (1915b). Prelude Op. 28 No. 7. In Mikuli, C., editor, *Complete Works for the Piano, Vol.9: Nocturnes*, page 10. G. Schirmer, Inc.

Chuan, C.-H. and Chew, E. (2007). A Dynamic Programming Approach to the Extraction of Phrase Boundaries from Tempo Variations in Expressive Performances. In *Proceedings of the 8th International Society for Music Information Retrieval Conference (ISMIR 2007)*, pages 305–308, Vienna, Austria.

Clark, A. (2013). Whatever next? Predictive brains, situated agents, and the future of cognitive science. *Behavioral and Brain Sciences*, 36(03):181–204.

Clarke, E. F. (1988). Generative principles in music. In Sloboda, J., editor, *Generative Processes in Music: The Psychology of Performance, Improvisation, and Composition*, pages 1–26. Oxford University Press.

Clarke, E. F. (1993). Imitating and evaluating real and transformed musical performances. *Music Perception*, 10(3):317–341.

Clynes, M. (1969). *Toward a Theory of Man: Precision of Essentic Form in Living Communication*, pages 177–206. Springer Berlin Heidelberg, Berlin, Heidelberg.

Clynes, M. (1986). Generative principles of musical thought integration of microstructure with structure. *Journal for the Integrated Study of Artificial Intelligence, Cognitive Science and Applied Epistemology*, 3(3):185–223.

Clynes, M. (1987). What can a musician learn about music performance from newly discovered microstruture principles (pm or pas)? In Gabrielsson, A., editor, *Action and Perception in Rhythm and Music*, volume 55, pages 201–233. Publications issued by the Royal Swedish Academy of Music, Stockholm, Sweden.

Clynes, M. (2005). Automatic Expressive Intonation Tuning System. U.S. Patent 6,924,426B2 issued August 2, 2005.

Conard, N. J. (2009). A female figurine from the basal Aurignacian of Hohle Fels Cave in southwestern Germany. *Nature*, 459(7244):248–252.

Conklin, D. (2013). Multiple Viewpoint Systems for Music Classification. *Journal of New Music Research*, 42(1):19–26.

Conklin, D. and Witten, I. H. (1995). Multiple viewpoint systems for music prediction. *Journal of New Music Research*, 24(1):51–73.

Cont, A. (2008). Antescofo: Anticipatory Synchronization and Control of Interactive Parameters in Computer Music. In *Proceedings of the 2008 International Computer Music Conference (ICMC 2008)*, pages 33–40, Belfast, Ireland.

Cont, A., Echeveste, J., Giavitto, J.-L., and Jacquemard, F. (2012). Correct Automatic Accompaniment Despite Machine Listening or Human Errors in Antescofo. In *Proceedings of the 38th International Computer Music Conference (ICMC 2012)*, Ljubljana, Slovenia.

Coorevits, E., Moelants, D., Maes, P.-J., and Leman, M. (2015). The influence of tempo on expressive timing: A multimodal approach. In *Proceedings of the Ninth Triennal Conference of the European Society for the Cognitive Sciences of Music (ESCOM 2015)*, pages 17–22, Manchester, Uk.

Coorevits, E., Moelants, D., Maes, P.-J., and Leman, M. (2017). Exploring the effect of tempo changes on violinists' body movements. *Musicae Scientiae*, pages 1–24.

Couperin, F. (1974). *The Art of Playing Harpsichord*. Alfred Music. Original work published 1716.

Dalla Bella, S. and Palmer, C. (2011). Rate Effects on Timing, Key Velocity, and Finger Kinematics in Piano Performance. *PLOS ONE*, 6(6):e20518.

Dannenberg, R. B. (1984). An On-Line Algorithm for Real-Time Accompaniment. In *Proceedings of the 1984 International Computer Music Conference*, pages 193–198, Paris, France.

Dannenberg, R. B., Gold, N. E., Liang, D., and Xia, G. (2014). Methods and prospects for human–computer performance of popular music. *Computer Music Journal*, 38(2):36–50.

Dannenberg, R. B. and Mohan, S. (2011). Characterizing Tempo Change in Musical Performances. In *Proceedings of the International Computer Music Conference (ICMC 2011)*, pages 650–656, Huddersfield, UK.

Davies, S. (1994). *Musical meaning and expression*. Cornell University Press.

Davies, S. (2001). Philosophical Perspectives on Music's Expressiveness. In Juslin, P. N. and Sloboda, J. A., editors, *Music and emotion: Theory and research.*, pages 23–44. Oxford University Press.

De Poli, G. (2004). Methodologies for Expressiveness Modelling of and for Music Performance. *Journal of New Music Research*, 33(3):189–202.

De Poli, G., Canazza, S., Rodà, A., and Schubert, E. (2014). The Role of Individual Difference in Judging Expressiveness of Computer-Assisted Music Performances by Experts. *ACM Transactions on Applied Perception*, 11(4):1–20.

Dempster, A. P., Laird, N. M., and Rubin, D. B. (1977). Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society. Series B (Methodological)*, 39(1):1–38.

Desain, P. and Honing, H. (1994). Does expressive timing in music performance scale proportionally with tempo? *Psychological Research*, 56:285–292.

Di Carlo, D. and Rodà, A. (2014). Automatic music "listening" for automatic music performance: a grandpiano dynamics classifier. In *Proceedings of the 1st International Workshop on Computer and Robotic Systems for Automatic Music Performance (SAMP 14)*, pages 1–8, Venice, Italy.

Dieleman, S., Schlüter, J., Raffel, C., Olson, E., Sønderby, S. K., Nouri, D., Maturana, D., Thoma, M., Battenberg, E., Kelly, J., Fauw, J. D., Heilman, M., de Almeida, D. M., McFee, B., Weideman, H., Takács, G., de Rivaz, P., Crall, J., Sanders, G., Rasul, K., Liu, C., French, G., and Degrave, J. (2015). Lasagne: First release.

Dixon, S., Goebl, W., and Cambouropoulos, E. (2006). Perceptual Smoothness of Tempo in Expressively Performed Music. *Music Perception*, 23(3):195–214.

Dixon, S., Goebl, W., and Widmer, G. (2005). The "Air Worm": an Interface for Real-Time manipulation of Expressive Music Performance. In *Proceedings of the 2005 International Computer Music Conference (ICMC 2005)*, Barcelona, Spain.

Dorfer, M., Henkel, F., and Widmer, G. (2018). Learning to Listen, Read, and Follow: Score Following as a Reinforcement Learning Game. In *In Proceedings of 19th International Society for Music Information Retrieval Conference (ISMIR 2018)*.

Durand, A. (2017). Probabilistic models for score following from symbolic and monophonic data. Rapport de stage d'ingénieur, TELECOM ParisTech and Department of Computational Perception, Johannes Kepler University Linz.

E-mu Systems, Inc. (2006). SoundFont Technical Specification. Technical report, E-mu Systems, Inc.

EBU-R-128 (2011). BU Tech 3341-2011, Practical Guidelines for Production and Implementation in Accordance with EBU R 128. https://tech.ebu.ch/docs/tech/tech3341.pdf.

Eerola, T., Friberg, A., and Bresin, R. (2013). Emotional expression in music: contribution, linearity, and additivity of primary musical cues. *Frontiers in Psychology*, 4:1–12.

Eitan, Z. and Granot, R. Y. (2006). How music moves: Musical parameters and listeners' images of motion. *Music Perception*, 23(3):221–247.

Elowsson, A. and Friberg, A. (2017). Predicting the perception of performed dynamics in music audio with ensemble learning. *The Journal of the Acoustical Society of America*, 141(3):2224–2242.

Fabian, D. (2017). Performing Music: Written traditions. In Ashley, R. and Timmers, R., editors, *The Routledge Companion to Music Cognition*. Routledge.

Fabiani, M. (2011). *Interactive computer-aided expressive music performance: Analysis, control, modification and synthesis*. PhD thesis, KTH Royal Institute of Technology.

Fabiani, M., Friberg, A., and Bresin, R. (2013). Systems for Interactive Control of Computer Generated Music Performance. In Kirke, A. and Miranda, E. R., editors, *Guide to Computing for Expressive Music Performance*, pages 49–73. Springer-Verlag, London, UK.

Farbood, M. M. (2012). A Parametric, Temporal Model of Musical Tension. *Music Perception*, 29(4):387–428.

Flossmann, S., Goebl, W., Grachten, M., Niedermayer, B., and Widmer, G. (2010). The Magaloff Project: An Interim Report. *Journal of New Music Research*, 39(4):363–377.

Flossmann, S., Grachten, M., and Widmer, G. (2011). Expressive Performance with Bayesian Networks and Linear Basis Models. In *Rencon Workshop Musical Performance Rendering Competition for Computer Systems (SMC-Rencon)*, Padova, Italy.

Flossmann, S., Grachten, M., and Widmer, G. (2013). Expressive Performance Rendering with Probabilistic Models. In Kirke, A. and Miranda, E. R., editors, *Guide to Computing for Expressive Music Performance*, pages 75–98. Springer, London, UK.

Fong, D. C.-L. and Saunders, M. (2011). LSMR: An Iterative Algorithm for Sparse Least-Squares Problems. *SIAM Journal on Scientific Computing*, 33(5):2950–2971.

Forth, J., Agres, K., Purver, M., and Wiggins, G. A. (2016). Entraining idyot: Timing in the information dynamics of thinking. *Frontiers in Psychology*, 7:1575.

Friberg, A. (1991). Generative Rules for Music Performance: A Formal Description of a Rule System. *Computer Music Journal*, 15 (2):56–71.

Friberg, A. (2005). Home conducting-control the Overall Musical expression with gestures. In *Proceedings of the 2005 International Computer Music Conference (ICMC 2005)*, Barcelona, Spain.

Friberg, A. (2006). pDM: An Expressive Sequencer with Real-Time Control of the KTH Music-Performance Rules. *Computer Music Journal*, 30(1):37–48.

Friberg, A. and Bisesi, E. (2014). Using Computational Models of Music Performance to Model Stylistic Variations. In Fabian, D., Timmers, R., and Schubert, E., editors, *Expressiveness in music performance: Empirical approaches across styles and cultures*, pages 240–259. Oxford University Press, Oxford, UK.

Friberg, A., Bresin, R., and Sundberg, J. (2006). Overview of the KTH rule system for musical performance. *Advances in Cognitive Psychology*, 2(2-3):145–161.

Friberg, A., Colombo, V., Frydén, L., and Sundberg, J. (2000). Generating Musical Performances with Director Musices. *Computer Music Journal*, 24(1):23–29.

Friberg, A. and Sundberg, J. (1999). Does music performance allude to locomotion? A model of final ritardandi derived from measurements of stopping runners. *Journal of the Acoustical Society of America*, 105(3):1469–1484.

Friston, K. and Kiebel, S. (2009). Predictive coding under the free-energy principle. *Philosophical transactions of the Royal Society of London. Series B, Biological sciences*, 364(1521):1211–1221.

Fu, M., Xia, G., Dannenberg, R., and Wasserman, L. (2015). A Statistical View on the Expressive Timing of Piano Rolled Chords. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR 2015)*, pages 578–584, Malaga, Spain.

Gabrielsson, A. (1974). Performance of rhythm patterns. *Scandinavian Journal of Psychology*, 15(1):63–72.

Gabrielsson, A. (1999). The Performance of Music. In Deutsch, D., editor, *The Psychology of Music*, pages 501–602. Academic Press, San Diego.

Gabrielsson, A. (2003). Music Performance Research at the Millennium. *Psychology of Music*, 31(3):221–272.

Gabrielsson, A., Bengtsson, I., and Gabrielsson, B. (1983). Performance of musical rhythm in 3/4 and 6/8 meter. *Scandinavian Journal of Psychology*, 24(1):193–213.

Gabrielsson, A. and Lindström, E. (2010). The role of structure in the musical expression of emotions. In *Handbook of music and emotion: Theory, research, applications*, pages 367–400. Oxford University Press.

Gadermaier, T., Grachten, M., and Cancino-Chacón, C. E. (2016). Basis-Function Modeling of Loudness Variations in Ensemble Performance. In *Proceedings of the 2nd International Conference on New Music Concepts (ICNMC 2016)*, Treviso, Italy.

Gasser, M., Arzt, A., Gadermaier, T., Grachten, M., and Widmer, G. (2015). Classical music on the web - user interfaces and data representations. In *Proceedings of the 16th International Society for Music Information Retrieval Conference, ISMIR 2015, Málaga, Spain, October 26-30, 2015*, pages 577–571.

Geringer, J. M., Clifford, K. M., McLeod, R., and Droe, K. (2006). The Effect of Articulation Style on Perception of Modulated Tempo. *Journal of Research in Music Education*, 54(4):324–336.

Gimpel, P. (2005). The Cat Concerto Controversy (Mistery Solved?). https://web.archive.org/web/20070928193003/http://www.gimpelmusicarchives.com/catconcerto.htm. Accessed: 13-04-2018.

Gingras, B., Pearce, M. T., Goodchild, M., Dean, R. T., Wiggins, G., and McAdams, S. (2016). Linking melodic expectation to expressive performance timing and perceived musical tension. *Journal of Experimental Psychology: Human Perception and Performance*, 42(4):594–609.

Giraldo, S. and Ramírez, R. (2016a). A machine learning approach to ornamentation modeling and synthesis in jazz guitar. *Journal of Mathematics and Music*, 10(2):107–126.

Giraldo, S. I. and Ramírez, R. (2016b). A Machine Learning Approach to Discover Rules for Expressive Performance Actions in Jazz Guitar Music. *Frontiers in Psychology*, 7:1965.

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*, pages 249–256.

Goebl, W. (1999). The Vienna 4x22 Piano Corpus.

Goebl, W. (2001). Melody lead in piano performance: Expressive device or artifact? *The Journal of the Acoustical Society of America*, 110(1):563–572.

Goebl, W. and Bresin, R. (2003). Measurement and reproduction accuracy of computer controlled grand pianos. *Journal of the Acoustical Society of America*, 114(4):2273–2283.

Goebl, W., Dixon, S., De Poli, G., Friberg, A., Bresin, R., and Widmer, G. (2008). 'Sense' in Expressive Music Performance: Data Acquisition, Computational Studies, and Models. In Polotti, P. and Rocchesso, D., editors, *Sound to Sense – Sense to Sound: A State of the Art in Sound and Music Computing*, pages 195–242. Logos, Berlin.

Goebl, W. and Palmer, C. (2009). Synchronization of Timing and Motion Among Performing Musicians. *Music Perception*, 26(5):427–438.

Goebl, W. and Widmer, G. (2009). On the Use of Computational Methods for Expressive Music Performance. In Crawford, T. and Gibson, L., editors, *Modern Methods for Musicology: Prospects, Proposals, and Realities*, pages 93–113. Ashgate, London.

Goodchild, M., Gingras, B., and McAdams, S. (2016). Analysis, Performance, and Tension Perception of an Unmeasured Prelude for Harpsichord. *Music Perception*, 34(1):1–20.

Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

Goto, M. (2007). Active music listening interfaces based on signal processing. In *Proceedings of the IEEE International Conference on Acoustics, Speech and Signal Processing 2007 (ICASSP 2007)*, pages 1441–1444, Honolulu, Hawaii, USA. IEEE.

Grachten, M. and Cancino Chacón, C. E. (2015). Strategies for Conceptual Change in Convolutional Neural Networks. Technical Report OFAI-TR-2015-04, Austrian Research Institute for Artificial Intelligence, Vienna, Austria.

Grachten, M. and Cancino-Chacón, C. E. (2017). Temporal dependencies in the expressive timing of classical piano performances. In Lessafre, M., Maes, P.-J., and Leman, M., editors, *The Routledge Companion to Embodied Music Interaction*, pages 360–369. Routledge.

Grachten, M., Cancino-Chacón, C. E., Gadermaier, T., and Widmer, G. (2017). Towards computer-assisted understanding of dynamics in symphonic music. *IEEE Multimedia*, 24(1):36–46.

Grachten, M., Cancino Chacón, C. E., and Widmer, G. (2014). Analysis and Prediction of Expressive Dynamics Using Bayesian Linear Models. In *Proceedings of the 1st International Workshop on Computer and Robotic Systems for Automatic Music Performance (SAMP 14)*, pages 545–552, Venice, Italy.

Grachten, M., Gasser, M., Arzt, A., and Widmer, G. (2013). Automatic alignment of music performances with structural differences. In *Proceedings of the 14th International Society for Music Information Retrieval Conference*, Curitiba, Brazil.

Grachten, M., Goebl, W., Flossmann, S., and Widmer, G. (2009). Phase-plane representation and visualization of gestural structure in expressive timing. *Journal of New Music Research*, 38(2):183–195.

Grachten, M. and Krebs, F. (2014). An Assessment of Learned Score Features for Modeling Expressive Dynamics in Music. *IEEE Transactions on Multimedia*, 16(5):1211–1218.

Grachten, M. and Widmer, G. (2009). Who is Who in the End? Recognizing Pianists by Their Final Ritardandi. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2009)*, pages 51–56, Kobe, Japan.

Grachten, M. and Widmer, G. (2012). Linear Basis Models for Prediction and Analysis of Musical Expression. *Journal of New Music Research*, 41(4):311–322.

Graves, A. (2013). Generating Sequences With Recurrent Neural Networks. *arXiv:1610.03606v1*, 1308:850.

Grindlay, G. and Helmbold, D. (2006). Modeling, analyzing, and synthesizing expressive piano performance with graphical models. *Machine Learning*, 65(2-3):361–387.

Gu, Y. and Raphael, C. (2012). Modeling Piano Interpretation Using Switching Kalman Filter. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR 2012)*, pages 145–150, Porto, Portugal.

Hamby, D. M. (1995). A comparison of sensitivity analyses techniques. *Health Physics*, 68(2):195–204.

Hashida, M., Matsui, T., and Katayose, H. (2008). A New Music Database Describing Deviation Information of Performance Expression. In *Proceedings of the 9th International Society for Music Information Retrieval Conference (ISMIR 2008)*, pages 489–495, Philadelphia, PA, USA.

Hashida, M., Nakamura, E., and Katayose, H. (2017). Constructing PEDB 2nd Edition: A Music Performance Database with Phrase Information. In *Proceedings of the 14th Sound and Music Computing Conference (SMC 2017)*, pages 359–364, Espoo, Finland.

Herremans, D. and Chew, E. (2016). Tension ribbons: Quantifying and visualising tonal tension. In *Proceedings of the Second International Conference on Technologies for Music Notation and Representation TENOR*, Cambridge, UK.

Herremans, D., Chuan, C.-H., and Chew, E. (2017). A Functional Taxonomy of Music Generation Systems. *ACM Computing Surveys*, 50(5):1–30.

Hill, J. (2012). Imagining Creativity: An Ethnomuscological Perspective on How Belief Systems Encourage or Inhibit Creative Activities in Music. In Hargreaves, D., Miell, D., and MacDonald, R., editors, *Musical Imaginations: Multidisciplinary Perspectives on Creativity, Performance, and Perception*. Oxford University Press.

Hiraga, R., Bresin, R., Hirata, K., and Katayose, H. (2003). After the first year of Rencon. In *Proceedings of the 2003 International Computer Music Conference, ICMC 2003, Singapore, September 29 - October 4, 2003*.

Hiraga, R., Bresin, R., Hirata, K., and Katayose, H. (2004). Rencon 2004: Turing Test for Musical Expression. In *Proceedings of the 2004 International Conference on New Interfaces for Musical Expression (NIME-04)*, pages 120–123, Hamamatsu, Japan.

Hiraga, R., Bresin, R., and Katayose, H. (2006). Rencon 2005. In *Proceedings of the 20th Annual Conference of the Japanese Society for Artificial Intelligence (JSAI2006)*, Funabori, Tokyo, Japan.

Hiraga, R., Hashida, M., Hirata, K., Katayose, H., and Noike, K. (2002). RENCON: toward a new evaluation system for performance rendering systems. In *Proceedings of the 2002 International Computer Music Conference (ICMC 2002)*, Gothenburg, Sweden.

Hochreiter, S. and Schmidhuber, J. (1997). Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780.

Hoffman, G. and Weinberg, G. (2011). Interactive improvisation with a robotic marimba player. *Autonomous Robots*, 31(2-3):133–153.

Hofmann, A., Chatziioannou, V., Weilguni, M., Goebl, W., and Kausel, W. (2013). Measurement

setup for articulatory transient differences in woodwind performance. *Proceedings of Meetings on Acoustics*, 19(1):035060.

Honing, H. (2001). From time to time: The representation of timing and tempo. *Computer Music Journal*, 35(3):50–61.

Honing, H. (2005). Timing is Tempo-Specific. In *Proceedings of the 2005 International Computer Music Conference (ICMC 2005)*, Barcelona, Spain.

Honing, H. (2006). Computational Modeling of Music Cognition: A Case Study on Model Selection. *Music Perception*, 23(5):365–376.

Hornik, K. (1991). Approximation capabilities of multilayer feedforward networks. *Neural Networks*, 4(2):251 – 257.

Humphrey, E. J., Bello, J. P., and LeCun, Y. (2012). Moving Beyond Feature Design: Deep Architectures and Automatic Feature Learning in Music Informatics. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR 2012)*, pages 403–408, Porto, Portugal.

Huron, D. (2006). *Sweet Anticipation*. MIT Press.

Huron, D. and Fantini, D. A. (1989). The avoidance of inner-voice entries: Perceptual evidence and musical practice. *Music Perception: An Interdisciplinary Journal*, 7(1):43–47.

Istók, E., Friberg, A., Huotilainen, M., and Tervaniemi, M. (2013). Expressive timing facilitates the neural processing of phrase boundaries in music: Evidence from event-related potentials. *PLoS ONE*, 8(1):1–11.

Juslin, P. (2003). Five facets of musical expression: a psychologist's perspective on music performance. *Psychology of Music*, 31(3):273–302.

Juslin, P. and Laukka, P. (2003). Communication of emotions in vocal expression and music performance: Different channels, same code? *Psychological Bulletin*, 129(5):770–814.

Juslin, P. N. (2001). Communicating emotion in music performance: a review and theoretical framework. In Juslin, P. N. and Sloboda, J. A., editors, *Music and Emotion: Theory and Research*, pages 309–337. Oxford University Press, Oxford, UK.

Juslin, P. N. and Sloboda, J. (2011). *Handbook of music and emotion: Theory, research, applications*. Oxford University Press, Oxford, UK.

Katayose, H., Hashida, M., De Poli, G., and Hirata, K. (2012). On Evaluating Systems for Generating Expressive Music Performance: the Rencon Experience. *Journal of New Music Research*, 41(4):299–310.

Keller, P. E., Knoblich, G., and Repp, B. (2007). Pianists duet better when they play with themselves: On the possible role of action simulation in synchronization. *Consciousness and Cognition*, 16:102–111.

Kendall, R. A. and Carterette, E. C. (1990). The Communication of Musical Expression. *Music Perception*, 8(2):129–163.

Kim, T. H., Fukayama, S., Nishimoto, T., and Sagayama, S. (2010). Performance rendering for polyphonic piano music with a combination of probabilistic models for melody and harmony. In *Proceedings of the 7th International Conference on Sound and Music Computing (SMC 2010)*, pages 23–30, Barcelona, Spain.

Kim, T. H., Fukayama, S., Nishimoto, T., and Sagayama, S. (2011). Polyhymnia: An automatic piano performance system with statistical modeling of polyphonic expression and musical symbol interpretation. In *Proceedings of the 11th International Conference on New Interfaces for Musical Expression (NIME 2011)*, pages 96–99, Oslo, Norway.

Kim, T. H., Fukuyama, S., Nishimoto, T., and Sagayama, S. (2013). Statistical Approach to Automatic Expressive Rendition of Polyphonic Piano Music. In Kirke, A. and Miranda, E. R., editors, *Guide to Computing for Expressive Music Performance*, pages 145–179. Springer, London, UK.

Kingma, D. and Ba, J. (2014). Adam: A Method for Stochastic Optimization. *arXiv:1412.6980*.

Kirke, A. and Miranda, E. R. (2013). An Overview of Computer Systems for Expressive Music Performance. In Kirke, A. and Miranda, E. R., editors, *Guide to Computing for Expressive Music Performance*, pages 1–48. Springer-Verlag, London, UK.

Kivy, P. (1991). *Music Alone: Philosophical Reflections on the Purely Musical Experience*. Cornell University Press.

Kosta, K., Bandtlow, O. F., and Chew, E. (2014). Practical Implications of Dynamic Markings in the Score: Is Piano Always Piano? In *Audio Engineering Society Conference: 53rd International Conference: Semantic Audio*, London, UK.

Kosta, K., Bandtlow, O. F., and Chew, E. (2015). A Change-Point Approach Towards Representing Musical Dynamics. In *Proceedings of the 5th International Conference on Mathematics and Computation in Music (MCM 2015)*, pages 179–184, London, UK.

Kosta, K., Ramírez, R., Bandtlow, O. F., and Chew, E. (2016). Mapping between dynamic markings and performed loudness: a machine learning approach. *Journal of Mathematics and Music*, 10(2):149–172.

Krebs, F. and Grachten, M. (2012). Combining score and filter based models to predict tempo fluctuations in expressive music performances. In *Proceedings of the Ninth Sound and Music Computing Conference (SMC 2012)*, Copenhagen, Denmark.

Krumhansl, C. L. (1990). *Cognitive foundations of musical pitch*. Cognitive foundations of musical pitch. Oxford University Press, New York.

Langner, J. and Goebl, W. (2003). Visualizing Expressive Performance in Tempo–Loudness Space. *Computer Music Journal*, 27(4):69–83.

Lattner, S., Cancino Chacón, C. E., and Grachten, M. (2015a). Pseudo-Supervised Training Improves Unsupervised Melody Segmentation. In *In Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence (IJCAI 2015)*, pages 2459–2465, Buenos Aires, Argentina.

Lattner, S., Grachten, M., Agres, K., and Cancino Chacón, C. E. (2015b). Probabilistic Segmentation of Musical Sequences using Restricted Boltzmann Machines. In *Fifth International Conference on Mathematics and Computation in Music (MCM 2015)*, London, UK.

Lem, S. (2002). *The Cyberiad*. Houghton Mifflin Hartcourt Publishing Company, New York, NY, Kindle edition.

Leman, M. (2008). *Embodied Music Cognition and Mediation Technology*. The MIT Press, Cambridge, MA.

Leman, M., Lesaffre, M., and Maes, P.-J. (2017a). Introduction: What is embodied Music

Interaction? In Lessafre, M., Maes, P.-J., and Leman, M., editors, *The Routledge Companion to Embodied Music Interaction*, pages 1–10. Routledge, New York, NY, USA.

Leman, M., Nijs, L., and Di Stefano, N. (2017b). On the Role of the Hand in the Expression of Music. In Bertolaso, M. and Di Stefano, N., editors, *The Hand: Perception, Cognition, Action*, pages 175–192. Springer International Publishing, Cham.

Lerdahl, F. and Jackendoff, R. (1983). *A Generative Theory of Tonal Music*. The MIT Press.

Lessafre, M., Maes, P.-J., and Leman, M., editors (2017). *The Routledge Companion to Embodied Music Interaction*. Routledge.

Levinson, J. (1998). *Music in the Moment*. Cornell University Press.

Li, S., Black, D., Chew, E., and Plumbley, M. D. (2014). Evidence that Phrase-Level Tempo Variation May be Represented Using a Limited Dictionary. In *Proceedings of the 13th International Conference for Music Perception and Cognition (ICMPC13-APSCOM5)*, pages 405–411, Seoul, South Korea.

Li, S., Black, D. A., and Plumbley, M. D. (2015). The clustering of expressive timing within a phrase in classical piano performances by Gaussian Mixture Models. In *Proceedings of the 11th International Symposium on Computer Music Multidisciplinary Research (CMMR 2015)*, pages 322–345, Plymouth, UK.

Li, S., Dixon, S., Black, D. A., and Plumbley, M. D. (2016). A model selection test on effective factors of the choice of expressive timing clusters for a phrase. In *Proceedings of the 13th Sound and Music Conference (SMC 2016)*, Hamburg, Germany.

Li, S., Dixon, S., and Plumbley, M. D. (2017). Clustering Expressive Timing with Regressed Polynomial Coefficients Demonstrated by a Model Selection Test. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR 2017)*, Suzhou, China.

Liebman, E., Ornoy, E., and Chor, B. (2012). A Phylogenetic Approach to Music Performance Analysis. *Journal of New Music Research*, 41(2):195–222.

Liem, C., Hanjalic, A., and Sapp, C. (2011). Expressivity in Musical Timing in Relation to Musical Structure and Interpretation: A Cross-Performance, Audio-Based Approach. In *Audio Engineering Society Conference: 42nd International Conference: Semantic Audio*, Ilmenau, Germany.

Liem, C. C., Gómez, E., and Schedl, M. (2015). PHENICX: Innovating the classical music experience. In *Proceedings of the 2015 IEEE International Conference on Multimedia & Expo Workshops (ICMEW)*, Turin, Italy. IEEE.

Liem, C. C. S. and Hanjalic, A. (2011). Expressive Timing from Cross-Performance and Audio-Based Alignment Patterns: An Extended Case Study. In *Proceedings of the 12th International Society for Music Information Retrieval (ISMIR 2011)*, pages 519–525, Miami, Fl, USA.

Liem, C. C. S. and Hanjalic, A. (2015). Comparative Analysis of Orchestral Performance Recordings: An Image-Based Approach. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR 2015)*, Malaga, Spain.

Lim, A., Mizumoto, T., Ogata, T., and Okuno, H. G. (2012). A musical robot that synchronizes with a coplayer using non-verbal cues. *Advanced Robotis*, 26:363–381.

Lindström, E. (1992). 5 x "oh, my darling clementine". the influence of expressive intention on music performance. Department of Psychology, Uppsala University.

Liszt, F. (1913). Ungarische Rhapsodie No. 2 S.244/2. In von Sauer, E., editor, *Klavierwerke, Band 1: Rhapsodien, Werke für Klavier zu 2 Händen*, pages 20–34. Edition Peters, Leipzig, Germany.

Longuet-Higgins, H. C. and Lee, C. S. (1982). The perception of musical rhythms. *Perception*, 11(2):115–128. PMID: 7155765.

Longuet-Higgins, H. C. and Lee, C. S. (1984). The rhythmic interpretation of monophonic music. *Music Perception: An Interdisciplinary Journal*, 1(4):424–441.

Maezawa, A. and Yamamoto, K. (2016). Automatic Music Accompaniment Based on Audio-Visual Score Following. In *Proceedings of the 17th International Society for Music Information Retrieval Conference Late-breakingDemo Session ISMIR 2017-LBD*, New York, NY, USA.

Marchini, M., Papiotis, P., and Maestre, E. (2013). Investigating the relationship between expressivity and synchronization in ensemble performance: An exploratory study. In *Proceedings of the International Symposium on Performance Science 2013 (ISPS 2013)*, pages 217–222, Vienna, Austria.

Marchini, M., Ramírez, R., Papiotis, P., and Maestre, E. (2014). The Sense of Ensemble: a Machine Learning Approach to Expressive Performance Modelling in String Quartets. *Journal of New Music Research*, 43(3):303–317.

Martens, J. (2010). Deep learning via Hessian-free optimization. In *Proceedings of the International Conference on Machine Learning (ICML-10)*, Haifa, Isrel.

Masko, J., Friberg, J. F., and Friberg, A. (2014). Software tools for automatic music performance. In *Proceedings of the 1st International Workshop on Computer and Robotic Systems for Automatic Music Performance (SAMP 14)*, pages 537–544, Venice, Italy.

McLeod, A. and Steedman, M. (2016). Hmm-based voice separation of midi performance. *Journal of New Music Research*, 45(1):17–26.

McPherson, G. E. and Welch, G. F., editors (2011). *The Oxford Handbook of Music Education*. Oxford University Press.

Meyer, L. B. (1961). *Emotion and Meaning in Music*. The University of Chicago Press.

Moelants, D., Demey, M., Grachten, M., Wu, C.-F., and Leman, M. (2012). The influence of an audience on performers: A comparison between rehearsal and concert using audio, video and movement data. *Journal of New Music Research*, 41(1):67–78.

Molina-Solana, M., Arcos, J.-L., and Gómez, E. (2008). Using Expressive Trends for Identifying Violin Performers. In *Proceedings of the 9th International Society for Music Information Retrieval Conference (ISMIR 2008)*, pages 495–500, Philadelphia, PA, USA.

Molina-Solana, M., Arcos, J.-L., and Gómez, E. (2010a). Identifying Violin Performers by their Expressive Trends. *Intelligent Data Analysis*, 14(5):555–571.

Molina-Solana, M., Grachten, M., and Widmer, G. (2010b). Evidence for Pianist Specific Rubato Style in Chopin Nocturnes. In *Proceedings of the 11th International Society for Music Information Retrieval (ISMIR 2010)*, Utrecht, The Netherlands.

Moog, R. A. and Rhea, T. L. (1990). Evolution of the Keyboard Interface: The Bösendorfer 290 SE Recording Piano and the Moog Multiply-Touch-Sensitive Keyboards. *Computer Music Journal*, 14(2):52–60.

Morgan, N. and Bourlard, H. (1990). Generalization and parameter estimation in feedforward nets: Some experiments. In Touretzky, D., editor, *Advances in Neural Information Processing Systems 2*, pages 630–637. Morgan-Kaufmann.

Moulieras, S. and Pachet, F. (2016). Maximum entropy models for generation of expressive music. *CoRR*, abs/1610.03606.

Mozart, L. (1787). *Gründlichen Violinschule*. Johann Jakob Lotter und Sohn, Ausburg, 3rd edition.

Murphy, K. P. (1998). Switching Kalman Filters. Technical Report 98-10, Compaq Cambridge Research Lab.

Nakamura, E., Cuvillier, P., Cont, A., Ono, N., and Sagayama, S. (2015a). Autoregressive Hidden Semi-Markov Model of Symbolic Music Performance For Score Following. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR 2015)*, Málaga, Spain.

Nakamura, E., Nakamura, T., Saito, Y., Ono, N., and Sagayama, S. (2014a). Outer-Product Hidden Markov Model and Polyphonci MIDI Score Following. *Journal of New Music Research*, 43(2):183–201.

Nakamura, E., Ono, N., and Sagayama, S. (2014b). Merged-Output HMM for Piano Fingering of Both Hands. In *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR 2014)*, pages 531–536, Taipei, Taiwan.

Nakamura, E., Ono, N., Sagayama, S., and Watanabe, K. (2015b). A Stochastic Temporal Model of Polyphonic MIDI Performance with Ornaments. *Journal of New Music Research*, 44(4):287–304.

Nakamura, E., Takeda, H., Yamamoto, R., Saito, Y., Sako, S., and Sagayama, S. (2013). Score Following Handling Performances with Arbitrary Repeats and Skips and Automatic Accompaniment. *Journal of Information Processing Society of Japan*, 54(4):1338–1349.

Nakamura, E., Yoshii, K., and Katayose, H. (2017). Performance Error Detection and Post-Processing for Fast and Accurate Symbolic Music Alignment. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR 2018)*, pages 347–353, Suzhou, China.

Narmour, E. (1990). *The analysis and cognition of basic melodic structures : the Implication-Realization model*. University of Chicago Press, Chicago, IL, USA.

Nocedal, J. and Wright, S. (2006). *Numerical Optimization*. Springer Series in Operations Research and Financial Engineering. Springer Science & Business Media.

Novembre, G. and Keller, P. E. (2014). A conceptual review on action-perception coupling in the musicians' brain: what is it good for? *Frontiers in Human Neuroscience*, 8(603):1–12.

Ockelford, A. (2006). Implication and expectation in music: a zygonic model. *Psychology of Music*, 34(1):81.

Ohishi, Y., Mochihashi, D., Kameoka, H., and Kashino, K. (2014). Mixture of Gaussian process experts for predicting sung melodic contour with expressive dynamic fluctuations. In *Proceedings of the 2014 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP 2014)*, pages 3714–3718, Florence, Italy. IEEE.

Okumura, K., Sako, S., and Kitamura, T. (2011). Stochastic Modeling of a Musical Performance with Expressive Representations from the Musical Score. In *Proceedings of the 12th Inter-*

national Society for Music Information Retrieval Conference (ISMIR 2011), pages 531–536, Miami, FL, USA.

Okumura, K., Sako, S., and Kitamura, T. (2014). Laminae: A stochastic modeling-based autonomous performance rendering system that elucidates performer characteristics. In *Joint Proceedings of the 40th International Computer Music Conference (ICMC 2014) and the 11th Sound and Music Computing Conference (SMC 2014)*, pages 1271–1276, Athens, Greece.

Paige, C. C. and Saunders, M. A. (1982). LSQR, An algorithm for sparse linear equations and sparse least squares. *ACM Trans. Math. Software*, 8(1):43–71.

Palmer, C. (1996). Anatomy of a Performance: Sources of Musical Expression. *Music Perception*, 13(3):433–453.

Palmer, C. (1997). Music Performance. *Annual Review of Psychology*, 48(1):115–138.

Pascanu, R., Mikolov, T., and Bengio, Y. (2013). On the difficulty of training recurrent neural networks. In *Proceedings of the 30th International Conference on Machine Learning*, pages 1–9, Atlanta, Georgia, USA.

Pearce, M., Müllensiefen, D., and Wiggins, G. A. (2008). A Comparison of Statistical and Rule-Based Models of Melodic Segmentation. In *Proceedings of the Ninth International Conference on Music Information Retrieval (ISMIR 2008)*, Philadelphia, PA, USA.

Pearce, M. T. (2005). *The Construction and Evaluation of Statistical Models of Melodic Structure in Music Perception and Composition*. PhD thesis, City University London, London UK.

Peng, L. and Gerhard, D. (2009). A Gestural Interface for Orchestral Conducting Education. In *Proceedings of the First International Conference on Computer Supported Education (CSEDU 2009)*, pages 406–409, Lisboa, Portugal.

Peperkamp, J., Hildebrandt, K., and Liem, C. C. S. (2017). A formalization of relative local tempo variations in collections of performances. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR 2017)*, Suzhou, China.

Petersen, K. B. and Pedersen, M. S. (2012). *The Matrix Cookbook* . Technical University of Denmark.

Platz, F. and Kopiez, R. (2012). When the Eye Listens: A Meta-analysis of How Audio-visual Presentation Enhances the Appreciation of Music Performance. *Music Perception*, 30(1):71–83.

Pylyshyn, Z. W. (1984). *Computation and Cognition: Toward a Foundation for Cognitive Science*. The MIT Press.

Rabiner, L. E. (1989). A Tutorial on Hidden Markov Models and Selected Applications in Speech Recognition. *Proceedings of the IEEE*, 77(2):257–286.

Ramírez, R., Maestre, E., Pertusa, A., Gómez, E., and Serra, X. (2007). Performance-based interpreter identification in saxophone audio recordings. *IEEE Transactions on Circuits and Systems for Video Technology*, 17(3):356–364.

Raphael, C. (2001a). Music Plus One: A System for Flexible and Expressive Musical Accompaniment. In *Proceedings of the 2001 International Computer Music Conference (ICMC 2001)*, Havana, Cuba.

Raphael, C. (2001b). Synthesizing musical accompaniments with Bayesian belief networks. *Journal of New Music Research*, 30(1):59–67.

Raphael, C. (2003). Orchestra in a box: A system for real-time musical accompaniment. In *IJCAI workshop program APP-5*, pages 5–10.

Raphael, C. (2009). Symbolic and Structural Representation of Melodic Expression. In *Proceedings of the 10th International Society for Music Information Retrieval Conference (ISMIR 2010)*, pages 555–560, Kobe, Japan.

Raphael, C. (2010). Music Plus One and Machine Learning. In *Proceedings of the 30th International Conference on Machine Learning (ICML 2010)*, Haifa, Israel.

Repp, B. H. (1992). Diversity and commonality in music performance - An analysis of timing microstructure in Schumann's "Träumerei". *Journal of the Acoustical Society of America*, 92(5):2546–2568.

Repp, B. H. (1994). Relational Invariance of Expressive Microstructure Across Global Tempo Changes in Music Performance: An Exploratory Study. *Psychological Research*, 56:285–292.

Repp, B. H. (1996). The art of inaccuracy: Why pianists' errors are difficult to hear. *Music Perception*, 14(2):161–184.

Repp, B. H. (1998). Obligatory "expectations" of expressive timing induced by perception of musical structure. *Psychological Research*, 61(1):33–43.

Repp, B. H. (2001). Processes underlying adaptation to tempo changes in sensorimotor synchronization. *Human Movement Science*, 20:277–312.

Repp, B. H., Windsor, L., and Desain, P. (2002). Effects of tempo on the timing of simple musical rhythms. *Music Perception*, 19(4):565–593.

Rink, J., editor (1995). *The Practice of Performance: Studies in Musical Interpretation*. Cambridge University Press, Cambridge Uk.

Rink, J., editor (2002). *Musical Performance. A Guide to Understanding*. Cambridge University Press, Cambridge UK.

Rink, J. (2003). In respect of performance: the view from musicology. *Psychology of Music*, 31(3):303–323.

Robertson, A. and Stevens, D., editors (1966). *Historia General de la Música*. Ediciones Istmo. Translated by Aníbal Froufe.

Rodà, A., Schubert, E., De Poli, G., and Canazza, S. (2015). Toward a musical Turing test for automatic music performance . In *International Symposium on Computer Music Multidisciplinary Research*.

Ross, B. C. (2014). Mutual Information between Discrete and Continuous Data Sets. *PLOS ONE*, 9(2):e87357–5.

Rowe, R. (1992). *Interactive Music Systems: Machine Listening and Composing*. MIT Press, Cambridge, MA, USA.

Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*, 323(9):533–536.

Russell, J. A. (1980). A Circumplex Model of Affect. *Journal of Personality and Social Psychology*, 39(6):1161–1178.

Saltelli, A., Annoni, P., Azzini, I., Campolongo, F., Ratto, M., and Tarantola, S. (2010). Variance based sensitivity analysis of model output. Design and estimator for the total sensitivity index. *Computer Physics Communications*, 181(2):259–270.

Sapp, C. S. (2007). Comparative Analysis of Multiple Musical Performances. In *Proceedings of the 8th International Society for Music Information Retrieval Conference (ISMIR 2007)*, Vienna, Austria.

Sapp, C. S. (2008). Hybrid Numeric/Rank Similarity Metrics for Musical Performance Analysis. In *Proceedings of the 9th International Society for Music Information Retrieval Conference (ISMIR 2008)*, pages 501–506, Philadelphia, PA, USA.

Sarasúa, Á., Melenhorst, M., Julià, C. F., and Gómez, E. (2016). Becoming the maestro - a game to enhance curiosity for classical music. In *Proceedings of the 8th International Conference on Games and Virtual Worlds for Serious Applications (VS-GAMES 2016)*, pages 1–4, Barcelona. IEEE.

Saunders, C., Hardoon, D. R., Shawe-Taylor, J., and Widmer, G. (2008). Using string kernels to identify famous performers from their playing style. *Intelligent Data Analysis*, 12(4):425–440.

Schäfer, A. M. and Zimmermann, H. G. (2006). Recurrent neural networks are universal approximators. In Kollias, S. D., Stafylopatis, A., Duch, W., and Oja, E., editors, *Artificial Neural Networks – ICANN 2006*, pages 632–640, Berlin, Heidelberg. Springer Berlin Heidelberg.

Schlüter, J. (2017). *Deep Learning for Event Detection, Sequence Labelling and Similarity Estimation in Music Signals*. PhD thesis, Johannes Kepler University Linz, Austria.

Schoonderwaldt, E. and Demoucron, M. (2009). Extraction of bowing parameters from violin performance combining motion capture and sensors. *The Journal of the Acoustical Society of America*, 126(5):2695.

Schubert, E., Canazza, S., De Poli, G., and Rodà, A. (2017). Algorithms can Mimic Human Piano Performance: The Deep Blues of Music. *Journal of New Music Research*, 46(2):175–186.

Schubert, E., De Poli, G., Rodà, A., and Canazza, S. (2014a). Music Systemisers and Music Empathisers – Do they rate expressiveness of computer generated performances the same? In *Joint Proceedings of the 40th International Computer Music Conference (ICMC 2014) and the 11th Sound and Music Computing Conference (SMC 2014)*, pages 223–227, Athens, Greece.

Schubert, E., Kreuz, G., and von Ossietzky, C. (2014b). Open ended descriptions of computer assisted interpretations of musical performance: An investigation of individual differences. In *Proceedings of the 1st International Workshop on Computer and Robotic Systems for Automatic Music Performance (SAMP 14)*, pages 565–573, Venice, Italy.

Schuster, M. and Paliwal, K. K. (1997). Bidirectional Recurrent Neural Networks. *IEEE Transactions on Signal Processing*, 45(11):2673–2681.

Sears, D. R. W. (2016). *The Classical Cadence as a Closing Schema: Learning, Memory, & Perception*. PhD thesis, McGill University, Montreal, Canada.

Seashore, C. E. (1938). *Psychology of Music*. McGraw-Hill Book Company, Inc., New York, NY, USA.

Sébastien, V., Ralambondrainy, H., Sébastien, O., and Conruyt, N. (2012). Score Analyzer: Automatically Determining Scores Difficulty Level for Instrumental e-Learning. In *Proceedings of the 13th International Society for Music Information Retrieval Conference (ISMIR 2012)*, pages 571–576, Porto, Portugal.

Simon, I., Oore, S., Dieleman, S., and Eck, D. (2017). Learning to Create Piano Performances. In *Proceedings of the NIPS 2017 Workshop on Machine Learning for Creativity and Design*, Long Beach, Ca, USA.

Sloboda, J. A. (1983). The communication of musical metre in piano performance. *Quarterly Journal of Experimental Psychology*, 35A:377–396.

Solis, J. and Takanishi, A. (2013). Anthropomorphic musical robots designed to produce physically embodied expressive performances of music. In Kirke, A. and Miranda, E. R., editors, *Guide to Computing for Expressive Music Performance*, pages 235–255. Springer.

Spiegel, M. R. and Liu, J. M. (1999). *Mathematical Handbook of Formulas and Tables*. Schaum's outlines. McGraw-Hill Companies.

Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., and Salakhutdinov, R. (2014). Dropout: A Simple Way to Prevent Neural Networks from Overfitting. *Journal of Machine Learning Research: Workshop and Conference Proceedings*, 2014(15):1929–1958.

Stamatatos, E. and Widmer, G. (2005). Automatic Identification of Music Performers with Learning Ensembles. *Artificial Intelligence*, 165(1):37–56.

Sundberg, J., Askenfelt, A., and Frydén, L. (1983). Musical Performance: A Synthesis-by-Rule Approach. *Computer Music Journal*, 7(1):37–43.

Sundberg, J., Friberg, A., and Bresin, R. (2003). Attempts to Reproduce a Pianist's Expressive Timing with Director Musices Performance Rules. *Journal of New Music Research*, 32(3):317–325.

Sundberg, J., Frydén, L., and Askenfelt, A. (1982). What tells yo the player is musical? A study of music performances by means of analysis-by synthesis. *KTH Speech Transmission Laboratory Quarterly Progress and Status Report*, 23:135–148.

Tekman, H. G. (2002). Perceptual integration of timing and intensity variations in the perception of musical accents. *The Journal of General Psychology*, 129:181–191.

Temperley, D. (2007). *Music and Probability*. MIT Press.

Teramura, K., Okuma, H., Taniguchi, Y., Makimoto, S., and Maeda, S. (2008). Gaussian process regression for rendering music performance. In *Proceedings of the 10th International Conference on Music Perception and Cognition (ICMPC 10)*, Sapporo, Japan.

Thompson, S., Williamon, A., and Valentine, E. (2007). Time-Dependent Characteristics of Performance Evaluation. *Music Perception*, 25(1):13–29.

Tieleman, T. and Hinton, G. (2012). Lecture 6.5-rmsprop: Divide the gradient by a running average of its recent magnitude. In *COURSERA: Neural Networks for Machine Learning*.

Tobudic, A. and Widmer, G. (2006). Relational IBL in Classical Music. *Machine Learning*, 64(3):5–24.

Todd, N. (1992). The Dynamics of Dynamics: A Model of Musical Expression. *Journal of the Acoustical Society of America*, 91:3540–3550.

Toiviainen, P., Luck, G., and Thompson, M. R. (2010). Embodied meter: hierarchical eigenmodes in music-induced movement. *Music Perception*, 28(1):59–70.

Tsay, C.-J. (2013). Sight over sound in the judgment of music performance. *Proceedings of the National Academy of Sciences of the United States of America*, 110(36):14580–14585.

Tschiatschek, S., Cancino Chacón, C. E., and Pernkopf, F. (2013). Bounds for Bayesian network classifiers with reduced precision parameters. In *Proceedings of the 2013 International Conference on Acoustics, Speech and Signal Processing (ICASSP 2013)*, pages 3357–3361, Vancouver, Canada. IEEE.

van Beethoven, L. (1862). Sonate No. 18 Op. 31 No. 3. In *Ludwig van Beethoven Werke, Serie 16: Sonaten für das Pianoforte*, pages 89–108. Breitkopf and Härtel, Leipzig, Germany.

van der Steen, M. C. and Keller, P. (2013). The ADaptation and Anticipation Model (ADAM) of sensorimotor synchronization. *Frontiers in Human Neuroscience*, 7.

van Herwaarden, S., Grachten, M., and de Haas, W. B. (2014). Predicting Expressive Dynamics in Piano Performances using Neural Networks. In *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR 2014)*, pages 47–52, Taipei, Taiwan.

van Kranenburg, P. and Backer, E. (2004). Musical style recognition - a quantitative approach. In *Proceedings of the Conference on Interdisciplinary Musicology*, Graz, Austria.

Velarde, G., Cancino Chacón, C., Meredith, D., Weyde, T., and Grachten, M. (2018). Convolution-based classification of audio and symbolic representations of music. *Journal of New Music Research*, 47(3):191–205. DOI 10.1080/09298215.2018.1458885.

Velarde, Gissel and Weyde, Tillman and Cancino Chacón, Carlos and Meredith, David and Grachten, Maarten (2016). Composer Recognition Based On 2D-Filtered Piano Rolls. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR 2016)*, pages 116–121, New York, NY, USA.

Vorberg, D. and Schulze, H. (2002). Linear phase-correction in synchronization: Predictions, parameter estimation, and simulations. *Journal of Mathematical Psychology*, 46.

Vos, J. and Rasch, R. (1981). The perceptual onset of musical tones. *Perception & Psychophysics*, 29(4):323–335.

Vovelle, M., editor (1986). *Historia Universal Moderna y Contemporánea I: La Edad Moderna Europea*. Salvat Editores, S. A., Barcelona, Spain.

Wang, Z., Bovik, A. C., Sheikh, H. R., and Simoncelli, E. P. (2004). Image quality assessment: from error visibility to structural similarity. *IEEE Transactions on Image Processing*, 13(4):600–612.

Wapnick, J., Campbell, L., Siddell-Strebel, J., and Darrow, A.-A. (2009). Effects of Non-Musical Attributes and Excerpt Duration on Ratings of High-Level Piano Performances. *Musicae Scientiae*, 13(1):35–54.

Wesolowski, B. C., Wind, S. A., and Engelhard, G. (2016). Examining Rater Precision in Music Performance Assessment: An Analysis of Rating Scale Structure Using the Multifaceted Rasch Partial Credit Model. *Music Perception*, 33(5):662–678.

Widmer, G. (1995). Modeling the Rational Basis of Musical Expression. *Computer Music Journal*, 19 (2):76–96.

Widmer, G. (1996). Learning Expressive Performance: The Structure-Level Approach. *Journal of New Music Research*, 25 (2):179–205.

Widmer, G. (2000). Large-scale Induction of Expressive Performance Rules: First Quantitative Results. In *Proceedings of the 2000 International Computer Music Conference (ICMC 2000)*, Berlin, Germany.

Widmer, G. (2003). Discovering simple rules in complex data: A meta-learning algorithm and some surprising musical discoveries. *Artificial Intelligence*, 146(2):129–148.

Widmer, G. (2017). Getting Closer to the Essence of Music: The *Con Espressione* Manifesto. *ACM Transactions on Intelligent Systems and Technology*, 8(2):1–13.

Widmer, G., Flossmann, S., and Grachten, M. (2009). YQX Plays Chopin. *AI Magazine*, 30(3):35–48.

Widmer, G. and Goebl, W. (2004). Computational Models of Expressive Music Performance: The State of the Art. *Journal of New Music Research*, 33(3):203–216.

Widmer, G. and Tobudic, A. (2002). Playing Mozart by Analogy: Learning Multi-Level Timing and Dynamics Strategies. *Journal of New Music Research*, 32:259–268.

Wiggins, G. A., Müllensiefen, D., and Pearce, M. T. (2010). On the non-existence of music: Why music theory is a figment of the imagination. *Musicae Scientiae*, $14(1_{suppl})$:231–255.

Wiggins, G. A., Pearce, M. T., and Müllensiefen, D. (2012). Computational Modeling of Music Cognition and Musical Creativity. In Dean, R. T., editor, *The Oxford Handbook of Computer Music*. Oxford University Press.

Williams, P. M. (1996). Using neural networks to model conditional multivariate densities. *Neural Computation*, 8(4):843–854.

Wu, Y., Zhang, S., Zhang, Y., Bengio, Y., and Salakhutdinov, R. (2016). On Multiplicative Integration with Recurrent Neural Networks. In *30th Conference on Neural Information Processing Systems (NIPS 2016)*, Barcelona, Spain.

Xia, G. (2016). *Expressive Collaborative Music Performance via Machine Learning*. PhD thesis, Carnegie Mellon University, Pittsburgh, PA.

Xia, G. and Dannenberg, R. B. (2015). Duet Interaction: Learning Musicianship for Automatic Accompaniment. In *Proceedings of the 15th International Conference on New Interfaces for Musical Expression (NIME 2015)*, Baton Rouge, LA, USA.

Xia, G., Wang, Y., Dannenberg, R., and Gordon, G. (2015). Spectral Learning for Expressive Interactive Ensemble Music Performance. In *Proceedings of the 16th International Society for Music Information Retrieval Conference (ISMIR 2015)*, pages 816–822, Málaga, Spain.

A Basis Functions

A.1 List of Basis Functions

In this section we provide the complete list of all basis functions defined in the main text. In this appendix, we take some descriptions of the basis functions directly from the main text (in particular for the features described in Chapter 6) for the sake of having a standalone document.

Following the notation introduced in Sections 2.3 and 3.2, $\mathfrak{P}_{\mathfrak{S}}$ denotes an expressive performance matched to score $\mathfrak{S}$, $\boldsymbol{\mathcal{X}} = \{n_0, \ldots, n_{N_x-1}\}$ represents the set of $|\boldsymbol{\mathcal{X}}| = N_x$ notes in a musical score and $\boldsymbol{\mathcal{O}} = \{o_0, \ldots, o_{N_o-1}\}$ represents the set of $|\boldsymbol{\mathcal{O}}| = N_o$ score onsets. The set of all notes that occur at the same score position as score note n_i will be denoted as $o(n_i) = \{n_j \in o \mid \text{onset}(n_j) = \text{onset}(n_i)\}$, with $\text{onset}(n_i)$ being the onset time of n_i. See Figure 3.4 for an example of a musical score and its different elements.

A.1.1 Pitch

1. **Polynomial pitch model**. A third order polynomial model to describe the dependency of an expressive parameter on pitch. This polynomial model defines three basis functions:

$$\varphi_{\text{pitch}}(n_i) = \frac{\text{pitch}(n_i)}{127} \quad \varphi_{\text{pitch}^2}(n_i) = \left(\frac{\text{pitch}(n_i)}{127}\right)^2 \quad \varphi_{\text{pitch}^3}(n_i) = \left(\frac{\text{pitch}(n_i)}{127}\right)^3, \tag{A.1}$$

   where $\text{pitch}(n_i)$ represents the MIDI note number of n_i.

2. **Highest, Lowest and Melody pitch**. Three features representing the chromatic pitch of the highest note, the lowest note, and the melody note at each onset.

$$\varphi_{\text{pitch}_h}(o_i) = \frac{\max \text{pitch}(o_i)}{127} \tag{A.2}$$

$$\varphi_{\text{pitch}_h}(o_i) = \frac{\min \text{pitch}(o_i)}{127} \tag{A.3}$$

$$\varphi_{\text{pitch}_m}(o_i) = \begin{cases} \frac{\text{pitch}(n_m)}{127} & \text{if } n_m \in o_i \text{ is a melody note} \\ 0 & \text{otherwise.} \end{cases} \tag{A.4}$$

   Note that the basis function representing the pitch of the melody requires a (manual) annotation of the main melody of the piece. In the datasets used in this work, only the Batik/Mozart has this property.

3. **Vertical interval class** Three features describing up to three vertical interval classes above the bass, i.e. the intervals between the notes of a chord and the lowest pitch, excluding pitch class repetition and octaves. The set of vertical interval classes with respect

to the lowest pitch is $\text{vintc}(o_i) = \{(\text{pitch}(n_j) - \min \text{pitch}(o_i)) \bmod 12 \mid n_j \in o_i\}$. We can express these basis functions as

$$\varphi_{vic_k}(o_i) = \begin{cases} \frac{1}{11} \text{uniquesort}(\text{vintc}(o_i), k+1) & \text{if } |\text{vintc}(o_i)| > k \\ 0 & \text{otherwise,} \end{cases} \tag{A.5}$$

where $\text{uniquesort}(\cdot, k)$ is a function that gets the unique elements of a set, orders their values in an ascending fashion and returns the k-th element.

4. **Piano-roll slice**. A set of 88 binary features describing the active notes at the current score position. Each feature is given by

$$\varphi_{\text{piano-roll}_j}(o_i) = \mathbb{1}\{j \in \text{active}(o_i)\}, \tag{A.6}$$

where $\mathbb{1}\{\cdot\}$ is the indicator function[1], and $\text{active}(o_i) = \{\text{pitch}(n_k) \mid n_k \in \mathfrak{S} \text{ and } n_k \text{ is active at onset}(o_i)\}$ are the pitches active at score position $\text{onset}(o_i)$.

5. **Vertical neighbors**. Two basis functions that evaluate to the number of simultaneous notes with lower, and higher pitches, respectively, and a third basis function that evaluates to the total number of simultaneous notes at that position.

$$\varphi_{\text{up. neighbors}}(n_i) = |\{n_j \in o(n_i) \mid \text{pitch}(n_j) > \text{pitch}(n_i)\}| \tag{A.7}$$
$$\varphi_{\text{low. neighbors}}(n_i) = |\{n_j \in o(n_i) \mid \text{pitch}(n_j) < \text{pitch}(n_i)\}| \tag{A.8}$$
$$\varphi_{\text{tot. neighbors}}(n_i) = |\{n_j \in o(n_i) \mid \text{pitch}(n_j) \neq \text{pitch}(n_i)\}| \tag{A.9}$$

A.1.2 Metrical

A time signature is represented as $\frac{a}{b}$, where a is the length of the bar in beats, and b is the beat type.

1. **Metrical**. Representation of the time signature of a piece, and the (metrical) position of each note in the bar. For each time signature $\frac{a}{b}$, there are $a+1$ basis functions: a basis functions indicate notes starting at each beat, respectively, and a single basis function indicates notes starting on a *weak* metrical position. For example, the basis function labeled $\frac{4}{4}$ *beat 1* evaluates to 1 for all notes that start on the first beat in a $\frac{4}{4}$ time signature, and to 0 otherwise.

$$\varphi_{\frac{a}{b}\ \text{beat}\ j}(n_i) = \mathbb{1}\left\{\frac{\text{onset}(n_i) \bmod a}{a} + 1 - j = 0\right\} \tag{A.10}$$
$$\varphi_{\frac{a}{b}\ \text{weak}}(n_i) = \mathbb{1}\left\{\frac{\text{onset}(n_i) \bmod a}{a} \notin \mathbb{Z}\right\} \tag{A.11}$$

2. **IOI**. The inter-onset-interval (IOI) is the time between the onsets. For each note in score onset o_i, a total of $2M_{\text{IOI}}$ basis functions represent the IOIs between the M_{IOI} previous onsets and the next M_{IOI} onsets, e.g. for $M_{\text{IOI}} = 3$, the onsets between $(i-2, i-3)$, $(i-1, i-2)$, $(i, i-1)$, $(i, i+1)$, $(i+1, i+2)$, and $(i+2, i+3)$. These basis functions provide some context of the (local) rhythmical structure of the music. Each of these basis functions are defined as

$$\varphi_{\text{IOI}\ (i\pm j, i\pm k)}(o_i) = \text{onset}(o_{i\pm k}) - \text{onset}(o_{i\pm j}) \tag{A.12}$$

[1] See Equation (H.19).

3. **Duration**. A basis function that encodes the duration of a note. While the IOI describes the time interval between two notes, the duration of a note refers to the time that such note is sounding.

$$\varphi_{\text{duration}}(n_i) = \text{duration}(n_i) \tag{A.13}$$

4. **Rest**. Indicates whether notes precede a rest.

$$\varphi_{\text{rest}}(n_i) = \mathbb{1}\left\{n_i \text{ is preceded by a rest}\right\}. \tag{A.14}$$

5. **Beat phase**. The relative location of an onset within the bar, computed as

$$\varphi_{b_\phi}(o_i) = \frac{\text{onset}(o_i) \bmod a}{a}, \tag{A.15}$$

where a is the length of the bar in beats.

6. **Metrical strength**. Three binary features encoding the metrical strength of the t-th onset. b_d is nonzero at the downbeat; b_s is nonzero at the secondary strong beat in duple meters (e.g. quarter-note 3 in $\frac{4}{4}$, and eighth-note 4 in $\frac{6}{8}$), and b_w is nonzero at weak metrical positions (i.e. whenever b_d and b_s are both zero).

$$\varphi_{b_d}(o_n) = \mathbb{1}\{\varphi_{b_\phi}(o_i) = 0\} \tag{A.16}$$
$$\varphi_{b_s}(o_n) = \mathbb{1}\{\text{onset}(o_n) \bmod a \in \text{secondary strong beat}\} \tag{A.17}$$
$$\varphi_{b_w}(o_n) = \mathbb{1}\{\varphi_{b_d}(o_n) = 0 \text{ and } \varphi_{b_s}(o_n) = 0\}. \tag{A.18}$$

A.1.3 Notated Performance Directives

In this section, we describe basis functions encoding notated performance directives. Figure A.1 shows the first 16 bars of Sonata No. 18 Op. 31 No. 3 by L. van Beethoven where the performance directives have been highlighted with colored rectangles. In this example, the marking ***p*** (highlighted in purple) appears three times (i.e. has three instances): in the first beat of the first bar, in the first beat of the seventh bar and in the first beat of the 16th bar. In the following discussion we will refer to the j-th instance of a performance marking as the j-th time that such a marking appears in the score in chronological order.

1. **Dynamics markings**. Bases that encode dynamics markings, such as shown in Figure 3.9. We have three groups of these functions:

   a) **Constant dynamics markings**. Basis functions that describe a constant relative loudness level, such as ***p*** and ***f*** (see the ***p*** markings highlighted in purple in the example in Figure A.1). For each constant dynamics marking cd, its corresponding basis function is defined as

$$\varphi_{cd}(n_i) = \mathbb{1}\{n_i \text{ is in a section with a } cd \text{ marking}\}. \tag{A.19}$$

   b) **Gradual dynamics markings**. Basis functions that describe gradual changes in loudness, such as ***crescendo*** and ***diminuendo*** (see the ***cresc.*** markings highlighted in green in Figure A.1), are represented through a combination of a ramp function, followed by a constant (step) function, that continues until a new constant dynamics marking (e.g. ***f***) appears, as illustrated by $\varphi_{\textbf{\textit{cresc}}}$ in Figure 3.9. The corresponding basis function for each type of gradual dynamics markings gd is given by

$$\varphi_{gd}(n_i) = \text{rampstep}_{gd}(n_i \mid \mathfrak{S}), \tag{A.20}$$

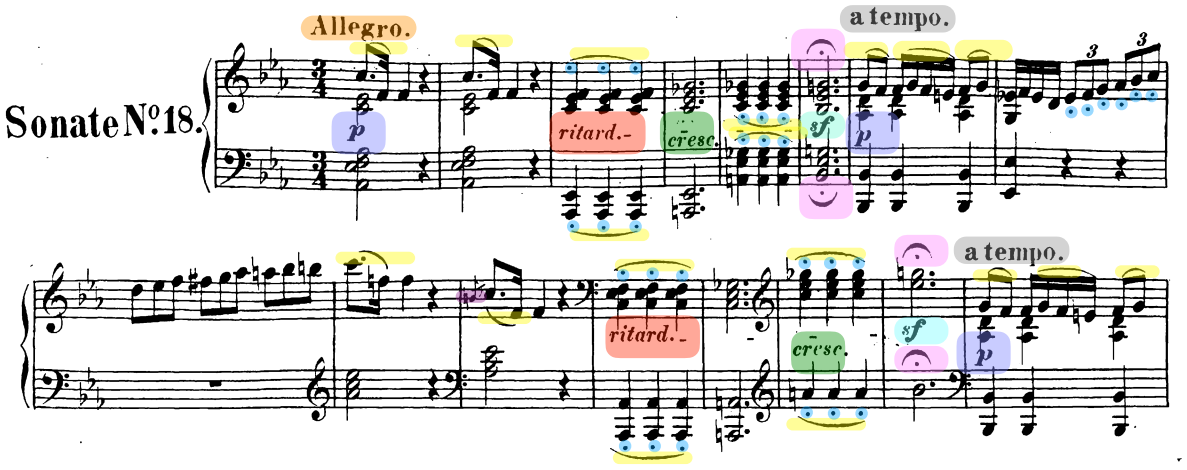


Figure A.1: Excerpt of the Sonata No. 18 Op. 31 No. 3 by L. van Beethoven. The colored rectangles/circles highlight notated performance directives. Purple rectangles highlight instances of ***p***; yellow rectangles highlight instances of legato slurs; green rectangles highlight instances of ***crescendo***, red rectangles highlight instances of ***ritardando*** (whose effect ends in the *a tempo* markings, highlighted in gray); little blue circles highlight staccato markings; pink rectangles highlight fermatas; light blue rectangles highlight instances of ***sf*** markings; an orange rectangle highlights an instance of **Allegro**; a violet rectangle highlights an instance of a grace note. The excerpt from the score comes from the Breitkopf & Härtel edition (van Beethoven, 1862).

where $\text{rampstep}_{gd}(\cdot)$ is a function given by

$$\text{rampstep}_{gd}(x_i \mid \mathfrak{S}) = \sum_{gd_j \in \mathfrak{S}} \text{rmst}_{gd_j}(n_i), \tag{A.21}$$

with $\text{rmst}_{gd_j}(\cdot)$ encoding the j-th instance in score $\mathfrak{S}$ of a gradual dynamics marking gd. This function is given as

$$\text{rmst}_{gd_j}(n_i) = \begin{cases} \frac{\text{onset}(n_i) - \text{start}(gd_j)}{\text{end}(gd_j) - \text{start}(gd_j)} & \text{if } \text{start}(gd_j) \leq \text{onset}(n_i) \leq \text{end}(gd_j) \\ 1 & \text{if } \text{end}(gd_j) < \text{onset}(n_i) < \text{next}(cd) \\ 0 & \text{otherwise} \end{cases} \tag{A.22}$$

where $\text{start}(gd_j)$ and $\text{end}(gd_j)$ represent the start and end time (in beats) of the marking, e.g. the beginning and end of the *hairpin*; and $\text{next}(cd)$ denotes the starting score position of the next constant dynamics marking.

c) **Indicator dynamics markings**. Basis functions that describe markings such as ***sf*** (*sforzato*) and ***sfz*** (*sforzando*) denoting an accented note (see the ***sf*** markings highlighted in light blue in Figure A.1). For each indicator dynamics marking id, its corresponding basis function is given as

$$\varphi_{id}(n_i) = \mathbb{1}\{n_i \text{ has an } id \text{ marking}\}. \tag{A.23}$$

2. **Context Dynamics Markings**.

a) **Gradual Dynamics Markings**. This set of basis functions, intends to differentiate between gradual loudness annotations in different loudness contexts. We do this

by combining each gradual annotation with its preceding and succeeding loudness level, for example ***p*** → ***crescendo*** → ***mf***, or ***f*** → ***diminuendo*** → ***mf***. For each combination $sd \to gd \to ed$ we define a function $\varphi_{sd\to gd\to ed}(n_i)$ computed using as

$$\varphi_{sd\to gd\to ed}(n_i) = \text{rampstep}_{sd\to gd\to ed}(n_i \mid \mathfrak{S}). \tag{A.24}$$

3. **Slur**. A representation of *legato* articulations indicating that musical notes are performed smoothly and connected, i.e. without silence between each note (see the legato slurs highlighted in yellow in Figure A.1). The beginning and ending of a slur are represented by decreasing and increasing ramp functions, respectively. The first (denoted *slur decr*) ranges from one to zero, while the second (denoted *slur incr*) ranges from zero to one over the course of the slur. These functions are given by

$$\varphi_{\text{slur incr}}(n_i) = \text{ramp}_{slur}(n_i \mid \mathfrak{S}) \tag{A.25}$$
$$\varphi_{\text{slur decr}}(n_i) = \text{decramp}_{slur}(n_i \mid \mathfrak{S}) \tag{A.26}$$

where $\text{ramp}_{slur}(\cdot), \text{decramp}_{slur}(\cdot)$ are functions given by

$$\text{ramp}_{slur}(n_i \mid \mathfrak{S}) = \sum_{slur_j \in \mathfrak{S}} \text{rmp}_{slur_j}(n_i) \tag{A.27}$$
$$\text{decramp}_{slur}(n_i \mid \mathfrak{S}) = \sum_{slur_j \in \mathfrak{S}} \text{rmdc}_{slur_j}(n_i) \tag{A.28}$$

with rmp_{slur_j} and dcrm_{slur_j} encoding the j-th instance in score $\mathfrak{S}$ of the legato slurs.

$$\text{rmp}_{slur_j}(n_i) = \begin{cases} \frac{\text{onset}(n_i)-\text{start}(slur_j)}{\text{end}(slur_j)-\text{start}(slur_j)} & \text{if start}(slur_j) \le \text{onset}(n_i) \le \text{end}(slur_j) \\ 0 & \text{otherwise} \end{cases} \tag{A.29}$$

$$\text{rmdc}_{slur_j}(n_i) = \begin{cases} \frac{\text{end}(slur_j)-\text{onset}(n_i)}{\text{end}(slur_j)-\text{start}(slur_j)} & \text{if start}(slur_j) \le \text{onset}(n_i) \le \text{end}(slur_j) \\ 0 & \text{otherwise} \end{cases} \tag{A.30}$$

4. **Tempo Markings**.

   a) **Constant tempo markings.** Bases that describe a constant tempo, such as **Adagio**, **Moderato** or **Allegro** (see the **Allegro** marking highlighted in orange in Figure A.1). For each *ct* dynamics marking, the basis functions are defined as

   $$\varphi_{ct}(o_i) = \mathbb{1}\{o_i \text{ is in a section with a } ct \text{ marking}\} \tag{A.31}$$

   b) **Gradual tempo markings** Encoding of markings that indicate gradual changes in the tempo of the music. These markings include ***rallentando***, ***ritardando*** and ***accelerando*** (see ***ritard.*** highlighted in red in Figure A.1). Mathematically, these basis functions can be defined in a similar way to the dynamics and markings. For each gradual tempo marking gt there is a basis function defined as

   $$\varphi_{gt}(o_i) = \text{ramp}_{gt}(o_i \mid \mathfrak{S}) \tag{A.32}$$

   where the start and end of the j-th instance of $gt \in \mathfrak{S}$ are and

   $$\text{start}(gt_j) = \text{onset}(gt_j)$$
   $$\text{end}(gt_j) = \begin{cases} \text{notated end of effect of } gt_j & \text{if dashes or } \textbf{\textit{a tempo}} \text{ are given} \\ \text{next}(ct) & \text{otherwise.} \end{cases}$$

5. **Repeat**. Takes into account repeat and ending barlines, i.e. explicit markings that indicate the structure of a piece by indicating the end of a particular section (which can be repeated), or the ending of a piece. The barlines are represented by an anticipating ramp function leading up to the repeat/ending barline over the course of a measure. For each type of barline bl (ending or repeat), we define a function

$$\varphi_{bl}(n_i) = \text{ramp}_{bl}(n_i \mid \mathfrak{S}, l_c), \tag{A.33}$$

where $\text{ramp}(n_i \mid \mathfrak{S}, l_c)$ is the ramp function defined in Equation (A.25) and $l_c \in \mathbb{R}_{>0}$ the length of the ramp, which defines the beginning and end of effect of the j-th instance of $bl \in \mathfrak{S}$ as $\text{start}(bl_j) = \text{onset}(bl_j) - l_c$ and $\text{end}(bl_j) = \text{onset}(bl_j)$. In our experiments, l_c is the length of the bar in beats.

6. **Accent**. Accents of individual notes or chords, such as the ***marcato*** in Figure 3.9. We define a basis function as

$$\varphi_{\text{accent}}(n_i) = \mathbb{1}\{n_i \text{ has an } at \text{ marking}\}. \tag{A.34}$$

7. **Staccato**. Encodes ***staccato*** markings on a note, an articulation indicating that a note should be temporally isolated from its successor, by shortening its duration (see the staccato markings highlighted in blue in Figure A.1). The basis function for these markings is defined as

$$\varphi_{\text{staccato}}(n_i) = \mathbb{1}\{n_i \text{ has a staccato marking}\}. \tag{A.35}$$

8. **Grace notes**. Encoding of musical ornaments that are melodically and or harmonically nonessential, but have an embellishment purpose (see the acciaccatura highlighted in violet in Figure A.1). The basis function for these ornaments is given as

$$\varphi_{\text{grace}}(n_i) = \mathbb{1}\{n_i \text{ is a grace note}\}. \tag{A.36}$$

9. **Fermata**. A basis function that encodes markings that indicate that a note should be prolonged beyond its normal duration (see the fermatas highlighted in pink in Figure A.1). A basis function encodes fermatas as

$$\varphi_{\text{fermata}}(n_i) = \mathbb{1}\{n_i \text{ has a fermata}\}. \tag{A.37}$$

A.1.4 Music Theoretic Features

1. **Harmonic** Two sets of indicator basis functions that encode a computer–generated harmonic analysis of the score based on the probabilistic polyphonic key identification algorithm proposed by Temperley (2007). This harmonic analysis produces an estimate of the key and scale degree, i.e. the roman numeral functional analysis of the harmony of the piece, for each bar of the score. A set of basis functions encode all major and minor keys while another set of basis functions encodes scale degrees. These basis functions are defined as

$$\varphi_{\text{key}_j}(n_i) = \mathbb{1}\{\text{key}(n_i) = j\} \tag{A.38}$$

$$\varphi_{\text{scale degree}_j}(n_i) = \mathbb{1}\{\text{scdeg}(n_i) = j\}, \tag{A.39}$$

where $\text{key}(n_i)$ is the key of the bar to which n_i belongs estimated by Temperley's HMM-based key estimation algorithm; and $\text{scdeg}(n_i)$ is the scale degree of the root of the tonic triad estimated by the key estimation method with respect to the global key of the piece. The key of the piece is taken from the score (if given), or using a global estimate using the key estimation algorithm.

A.1.5 Expectancy Features

IDyOM provides a conditional probability distribution of a musical event v, given a preceding sequence of events, i.e. $p(v_i \mid v_{i-1}, v_{i-2}, \dots)$. These events are represented as *viewpoints*, following the multiple viewpoint system (Conklin and Witten, 1995). We denote $v(x_i)$ to the viewpoints representing element x_i in the score.

1. **Information content** (IC). The IC measures the unexpectedness of a musical event, and is computed as

$$IC(v(x_i)) = -\log_2 p(v(x_i) \mid v(x_{i-1}), v(x_{i-2}), \dots). \tag{A.40}$$

   a) IC_m. The information content for each melody note. This value is computed using a model that is trained to predict the next chromatic melody pitch using a selection of melodic viewpoints, such as pitch interval (i.e. the arithmetic difference between two consecutive chromatic pitches, measured in MIDI note values), and contour (whether the chromatic pitch sequence rises, falls or remains the same). IDyOM performs a stepwise selection procedure that combines viewpoint models if they minimize model uncertainty as measured by corpus cross entropy (Sears, 2016, pp. 255-275).

$$\varphi_{IC_m}(o_i) = \begin{cases} \frac{1}{Z_{IC_m}} IC(v_{\text{melody}}(n_m)) & \text{if } n_m \in o_i \text{ is a melody note} \\ \varphi_{IC_m}(o_{i-1}) & \text{otherwise} \end{cases} \tag{A.41}$$

   where v_{melody} is the viewpoint representation of the melody notes and Z_{IC_m} is a normalization constant (see Section 6.3.2).

   b) IC_c. Estimation of the IC computed for the combination of pitch events (a proxy for harmony) at each score onset. IDyOM predicts the next combination of vertical interval classes above the bass.

$$\varphi_{IC_c}(o_i) = \frac{1}{Z_{IC_c}} IC(v_{\text{chord}}(o_i)). \tag{A.42}$$

   where v_{chord} uses vertical interval classes to represent onset o_i and Z_{IC_c} is a normalization constant computed in the same way as Z_{IC_m}.

2. **Entropy** is a measure of the degree of choice or uncertainty associated with a predicted outcome. The entropy can be computed as

$$H(v_i) = \mathbb{E}\{-\log_2 p(v_i \mid v_{i-1}, v_{i-2}, \dots)\} \tag{A.43}$$

   a) H_m. Entropy computed for each chromatic pitch in the melody. This is computed in the same fashion as the information content for melody notes, i.e.

$$\varphi_{H_m}(o_i) = \begin{cases} \frac{1}{Z_{H_m}} H(v_{\text{melody}}(n_m)) & \text{if } n_m \in o_i \text{ is a melody note} \\ \varphi_{H_m}(o_{i-1}) & \text{otherwise} \end{cases} \tag{A.44}$$

   where Z_{H_m} is a normalization constant computed in the same way as Z_{IC_m}.

   b) H_c. Entropy computed for the combined pitch events at each score onset.

$$\varphi_{H_c}(o_i) = \frac{1}{Z_{H_c}} H(v_{\text{chord}}(o_i)) \tag{A.45}$$

   where Z_{H_c} is a normalization constant computed in the same way as Z_{IC_m}.

A.1.6 Tonal Tension Features

Chew's spiral array is a parametric helix in $\mathbb{R}^3$ represented by

$$\mathrm{sa}(n_i) = \begin{pmatrix} \sin(u(n_i)) \\ \cos(u(n_i)) \\ \frac{\pi}{\sqrt{30}} u(n_i) \end{pmatrix}, \tag{A.46}$$

where $u\colon \boldsymbol{\mathcal{X}} \mapsto \mathbb{R}$ is a function denoting the position of the non-enharmonically equivalent pitch classes (e.g. C♯ and D♭ do not map into the same point) in the circle of fifths given by

$$u(n_i) = \frac{\pi}{2}\mathrm{cof}(n_i), \tag{A.47}$$

where $\mathrm{cof}(n_i)$ represents the (signed) number of fifths that the pitch class of n_i is away from C such that notes above and below C having positive and negative values, respectively (e.g. $\mathrm{cof}(\mathrm{C}) = 0$, $\mathrm{cof}(\mathrm{B}\flat) = -2$ and $\mathrm{cof}(\mathrm{G}) = 1$)[2].

Herremans and Chew (2016) propose using a sliding window approach, since musical tension varies over time. A cloud is defined as the set of points in the spiral array corresponding to one such windows. In this work, we define the cloud centered on onset o_i as all active notes within a beat from the center of the cloud i.e.

$$\mathrm{cloud}(o_i) = \{\mathrm{sa}(n_j) \mid n_j \text{ is active during } [\mathrm{onset}(o_i) - 1, \mathrm{onset}(o_i) + 1]\}. \tag{A.48}$$

The *center of effect* of a cloud is a point that represents the tonal center of the cloud, computed as

$$\mathrm{coe}(\mathrm{cloud}(o_i)) = \frac{\sum_{n_j \in \mathrm{cloud}(o_i)} \mathrm{duration}(n_j)\mathrm{sa}(n_j)}{\sum_{n_j \in \mathrm{cloud}(o_i)} \mathrm{duration}(n_j)}. \tag{A.49}$$

The tonal tension features are:

1. **Cloud diameter** (T_{cd}), which estimates the maximal tonal distance between notes in a segment of music, computed as

$$\varphi_{T_{cd}} = \frac{1}{Z_T} \max \mathrm{pairwisedist}(\mathrm{cloud}(o_i)) \tag{A.50}$$

   where pairwisedist is a function that computes the euclidean distance between all points in the cloud, and Z_T is a normalization constant.

2. **Cloud momentum** (T_{cm}) quantifies harmonic movement as the tonal distance from a segment of music to the next:

$$\varphi_{T_{cm}}(o_i) = \begin{cases} \frac{1}{Z_T}\|\mathrm{coe}(\mathrm{cloud}(o_i)) - \mathrm{coe}(\mathrm{cloud}(o_{i-1}))\| & i > 0 \\ 0 & i = 0 \end{cases} \tag{A.51}$$

3. **Tensile strain** (T_{ts}), the relative tonal distance between the current segment and the center of effect of the key of the piece.

$$\varphi_{T_{ts}}(o_i) = \frac{1}{Z_T}\|\mathrm{coe}(\mathrm{cloud}(o_i)) - \mathrm{coe}_{\mathrm{key}}(o_i))\|, \tag{A.52}$$

   where $\mathrm{coe}_{\mathrm{key}}(o_i)$ is the center of effect of the key, computed using the method described in (Chew, 2000).

[2] Note that the selection of C as the center of this representation is arbitrary, and a similar representation can be described around any note.

A.2 List of Orchestral Basis Functions

This appendix lists the basis functions in the RCO/Symphonic dataset. Each group lists the different categories of basis functions.

Accent

1. Accent.

   A Total of 33 Basis functions, each corresponding to an instrument listed in Appendix A.3.

Duration

1. Duration.

   A Total of 33 Basis functions, each corresponding to an instrument listed in Appendix A.3.

Fermata

1. Fermata.

   A Total of 33 Basis functions, each corresponding to an instrument listed in Appendix A.3.

IOI

1. $(i,\, i-1)$.

   A Total of 33 Basis functions, each corresponding to an instrument listed in Appendix A.3.

Metrical

As previously explained, there is a basis function per beat per instrument, e.g. Violin $\frac{4}{4}$ beat 1 and Violin $\frac{4}{4}$ beat 2 are two different basis functions.

1. $\frac{3}{8}$: {beat 1, beat 2, beat 3, weak}

   A total of 67 basis functions corresponding to the following instruments

   - Viola
   - Timpani
   - Bass Trombone
   - Contrabass Tuba
   - Bassoon
   - Bass Drum
   - Trumpet
   - Cello
   - Clarinet
   - Violin
   - Oboe
   - Flute

- Bass Clarinet
- Tam-Tam
- Contrabass
- Triangle
- Alto Trombone
- English Horn
- Glockenspiel
- Cymbals
- Harp
- French Horn

2. $\frac{2}{4}$: {beat 1, beat 2, weak}

   A total of 50 basis functions corresponding to the following instruments:

   - Oboe
   - Triangle
   - Harp
   - English Horn
   - Glockenspiel
   - Cello
   - Violin
   - Contrabassoon
   - Bassoon
   - Soprano Voice
   - Timpani
   - Bass Trombone
   - Viola
   - Contrabass
   - Sleigh Bells
   - Flute
   - Clarinet
   - French Horn
   - Trumpet

3. $\frac{3}{4}$: {beat 1, beat 2, beat 3, weak}

   A total of 85 basis functions corresponding to the following instruments:

   - Flute
   - Timpani
   - Tenor Voice
   - Trombone
   - Contrabassoon
   - Piccolo
   - Contrabass
   - Alto Trombone
   - Cello
   - Clarinet
   - Alto Voice
   - Viola
   - Soprano Voice
   - Bass Trombone
   - Contrabass Tuba
   - Bassoon
   - Trumpet
   - Violin
   - Oboe
   - Bass Voice
   - French Horn

4. $\frac{6}{8}$: {beat 1, beat 2, beat 3, beat 4, beat 5, beat 6, weak}

   A total of 103 basis functions corresponding to the follow instruments

- Violin
- Timpani
- Cello
- Tenor Voice
- Baritone Voice
- Contrabass
- Piccolo
- Flute
- Clarinet
- Trombone
- Alto Voice
- Bassoon
- Soprano Voice
- Trumpet
- Bass Voice
- Viola
- Contrabassoon
- French Horn
- Oboe

5. $\frac{4}{4}$: {beat 1, beat 2, beat 3, beat 4, weak}

   A total of 156 basis functions, corresponding to the follow instruments

- Bass Voice
- Wagner Tuba
- Contrabass Tuba
- Soprano Voice
- Bassoon
- Contrabass
- Flute
- Trombone
- Cymbals
- Viola
- Clarinet
- Bass Clarinet
- Oboe
- French Horn
- Bass Drum
- Tenor Voice
- Baritone Voice
- Triangle
- English Horn
- Piccolo
- Cello
- Contrabassoon
- Alto Trombone
- Glockenspiel
- Trumpet
- Harp
- Alto Voice
- Tam-Tam
- Sleigh Bells
- Violin
- Timpani
- Bass Trombone

6. $\frac{2}{2}$: {beat 1, beat 2, weak}

   A total of 71 basis functions, corresponding to the following instruments:

- Oboe
- French Horn
- Baritone Voice
- Bass Clarinet

- Viola
- Piccolo
- Trumpet
- Tenor Voice
- Alto Trombone
- Bass Voice
- Contrabass
- Violin
- Contrabassoon
- Contrabass Tuba
- Tenor Trombone
- Bassoon
- Trombone
- Soprano Voice
- Timpani
- Cello
- Alto Voice
- Bass Trombone
- Flute
- Clarinet

7. $\frac{12}{8}$: {beat 0, beat 1, beat 2, beat 3, beat 4, beat 5, beat 6, beat 7, beat 8, beat 9, beat 10, beat 11, weak}

   A total of 140 basis functions, corresponding to the following instruments:

- French Horn
- Oboe
- Violin
- Clarinet
- Flute
- Timpani
- Contrabass
- Cello
- Bassoon
- Trumpet
- Viola

Onset Indicator

1. Onset.

   A Total of 33 Basis functions, each corresponding an instrument listed in Appendix A.3.

Polynomial Pitch

1. Pitch.

   A Total of 33 Basis functions, each corresponding an instrument listed in Appendix A.3.

Repeat

1. Repeat end wide.

   A Total of 33 Basis functions, each corresponding an instrument listed in Appendix A.3.

Ritardando

1. Ritardando

   A Total of 33 Basis functions, each corresponding an instrument listed in Appendix A.3.

Slur

1. Slur increase

   A Total of 33 Basis functions, each corresponding an instrument listed in Appendix A.3.

2. Slur decrease

   A Total of 33 Basis functions, each corresponding an instrument listed in Appendix A.3.

3. Slur step

   A Total of 33 Basis functions, each corresponding an instrument listed in Appendix A.3.

Staccato

1. Staccato

   A Total of 33 Basis functions, each corresponding an instrument listed in Appendix A.3.

Unique Loudness Annotations

1. ***pppp***

   A total of 9 basis functions corresponding to the following instruments:

   - Flute
   - Harp
   - Contrabass
   - Clarinet
   - Bass Drum
   - Timpani
   - Viola
   - Violin
   - Cello

2. ***ppp***

   A total of 20 basis functions corresponding to the following instruments

   - Bass Drum
   - Cello
   - French Horn
   - Trumpet
   - Oboe
   - Contrabass Tuba
   - Timpani
   - Clarinet
   - Bass Trombone
   - Violin
   - Bass Clarinet
   - English Horn

- Bassoon
- Viola
- Flute
- Alto Trombone
- Contrabass
- Harp
- Cymbals
- Sleigh Bells

3. ***pp***

   A total of 31 basis functions corresponding to the following instruments

   - Harp
   - Triangle
   - Tenor Trombone
   - Bass Drum
   - Tam-Tam
   - Sleigh Bells
   - Trombone
   - Contrabassoon
   - Soprano Voice
   - Clarinet
   - Violin
   - Contrabass
   - Wagner Tuba
   - Tenor Voice
   - Cello
   - Flute
   - Trumpet
   - Bass Clarinet
   - Contrabass Tuba
   - Bass Voice
   - Viola
   - English Horn
   - Bass Trombone
   - Alto Voice
   - Alto Trombone
   - Oboe
   - Cymbals
   - Piccolo
   - Bassoon
   - Timpani
   - French Horn

4. ***p***

   A total of 31 basis functions corresponding to the following instruments

   - Timpani
   - Bassoon
   - Sleigh Bells
   - Violin
   - Trombone
   - Alto Voice
   - Clarinet
   - Contrabass Tuba
   - Contrabass
   - Contrabassoon
   - French Horn
   - Oboe
   - Cello
   - Viola
   - Bass Voice
   - Bass Clarinet

- Tenor Trombone
- Glockenspiel
- Trumpet
- Tenor Voice
- Harp
- Bass Trombone
- Piccolo
- English Horn
- Bass Drum
- Soprano Voice
- Alto Trombone
- Wagner Tuba
- Flute
- Cymbals
- Triangle

5. ***mp***

   A total of 7 basis functions corresponding to the following instruments

   - Violin
   - Bassoon
   - Bass Clarinet
   - Oboe
   - French Horn
   - Clarinet
   - Trumpet

6. ***mf***

   A total of 23 basis functions corresponding to the following instruments

   - Sleigh Bells
   - Bass Trombone
   - English Horn
   - Alto Trombone
   - Violin
   - Cymbals
   - Soprano Voice
   - Bassoon
   - Bass Clarinet
   - Contrabass
   - Viola
   - Tam-Tam
   - Timpani
   - Oboe
   - Bass Drum
   - Trumpet
   - Flute
   - Cello
   - Contrabass Tuba
   - Wagner Tuba
   - French Horn
   - Clarinet
   - Triangle

7. ***f***

   A total of 32 basis functions corresponding to the following instruments:

- Timpani
- Bassoon
- Sleigh Bells
- Violin
- Trombone
- Baritone Voice
- Alto Voice
- Clarinet
- Contrabass Tuba
- Contrabass
- Contrabassoon
- French Horn
- Cello
- Oboe
- Viola
- Bass Voice
- Bass Clarinet
- Tenor Trombone
- Glockenspiel
- Trumpet
- Tenor Voice
- Cymbals
- Harp
- Bass Trombone
- Piccolo
- English Horn
- Bass Drum
- Soprano Voice
- Alto Trombone
- Wagner Tuba
- Flute
- Triangle

8. ***ff***

   A total of 31 basis functions corresponding to the following instruments:

   - Contrabassoon
   - Contrabass Tuba
   - Cello
   - Trumpet
   - Glockenspiel
   - Wagner Tuba
   - French Horn
   - Harp
   - Cymbals
   - Flute
   - Clarinet
   - English Horn
   - Trombone
   - Violin
   - Alto Trombone
   - Tenor Voice
   - Triangle
   - Piccolo
   - Bass Clarinet
   - Bass Voice
   - Bass Trombone
   - Oboe
   - Alto Voice
   - Tam-Tam
   - Contrabass
   - Baritone Voice
   - Bassoon
   - Soprano Voice
   - Bass Drum
   - Viola
   - Timpani

9. ***fff***

   A total of 16 basis functions corresponding to the following instruments:

   - Wagner Tuba
   - Flute
   - Contrabass Tuba
   - Bass Trombone
   - Harp
   - Timpani
   - Contrabass
   - Cello
   - Bassoon
   - Clarinet
   - Oboe
   - Trumpet
   - French Horn
   - Violin
   - Alto Trombone
   - Viola

10. ***fp***

    A total of 15 basis functions corresponding to the following instruments

    - Oboe
    - Cello
    - French Horn
    - Flute
    - Clarinet
    - English Horn
    - Contrabassoon
    - Violin
    - Piccolo
    - Viola
    - Bass Trombone
    - Trumpet
    - Contrabass
    - Bassoon
    - Timpani

11. ***sf***

    A total of 22 basis functions corresponding to the following instruments

    - Bassoon
    - Baritone Voice
    - French Horn
    - Timpani
    - Contrabass
    - Tenor Voice
    - Soprano Voice
    - Contrabassoon
    - Violin
    - Oboe
    - Bass Clarinet
    - Bass Voice
    - Alto Voice
    - English Horn
    - Flute
    - Clarinet
    - Cello
    - Trumpet
    - Piccolo
    - Viola

- Trombone
- Harp

12. ***sfp***

    A total of 6 basis functions corresponding to the following instruments

    - Bassoon
    - Oboe
    - Violin
    - French Horn
    - Viola
    - Trumpet

13. ***crescendo***

    A total of 28 basis functions corresponding to the following instruments:

    - Flute
    - English Horn
    - Alto Trombone
    - Piccolo
    - Contrabassoon
    - Timpani
    - Bass Voice
    - Wagner Tuba
    - Bassoon
    - Cello
    - Sleigh Bells
    - Clarinet
    - Bass Drum
    - Oboe
    - Bass Clarinet
    - Baritone Voice
    - Contrabass Tuba
    - Triangle
    - Soprano Voice
    - Alto Voice
    - Contrabass
    - Trumpet
    - Violin
    - Tenor Voice
    - Bass Trombone
    - Viola
    - Harp
    - French Horn

14. ***diminuendo***

    A total of 29 basis functions corresponding to the following instruments:

    - Contrabass Tuba
    - Bass Clarinet
    - Trumpet
    - Trombone
    - Violin
    - Oboe
    - Viola
    - Bass Voice
    - Harp
    - Cymbals
    - Tenor Voice
    - Timpani
    - Alto Voice
    - Clarinet

- Sleigh Bells
- Cello
- Bass Drum
- Wagner Tuba
- Contrabass
- Soprano Voice
- Triangle
- Piccolo
- Baritone Voice
- Flute
- Bassoon
- Alto Trombone
- French Horn
- Bass Trombone
- English Horn

15. *sotto voce*

    A total of 4 basis functions corresponding to the following instruments:

    - Violin
    - Cello
    - Viola
    - Contrabass

16. *espressivo*

    A total of 6 basis functions corresponding to the following instruments:

    - Violin
    - Oboe
    - Flute
    - Bassoon
    - Viola
    - Clarinet

17. *dolce*

    A total of 9 basis functions corresponding to the following instruments:

    - Cello
    - Violin
    - Oboe
    - Bassoon
    - Flute
    - French Horn
    - Clarinet
    - Viola
    - Piccolo

Vertical Neighbor

1. Total neighbors

   A Total of 33 Basis functions, each corresponding an instrument listed in Appendix A.3.

A.3 Instrument List

This appendix provides a list of the instruments present in the RCO/Symphonic dataset.

Strings

- Violin
- Viola
- Violoncello
- Contrabass

Woodwinds

- Piccolo
- Flute
- Clarinet
- English Horn
- Oboe
- Bass Clarinet
- Bassoon
- Contrabassoon

Brass

- Trumpet
- French Horn
- Trombone
- Alto Trombone
- Tenor Trombone
- Bass Trombone
- Wagner Tuba
- Contrabass Tuba

Pitched and Unpitched Percussion

- Triangle
- Sleigh Bells
- Glockenspiel
- Cymbals
- Timpani
- Tam-Tam
- Bass Drum

Plucked Strings

- Harp

Voice

- Soprano
- Alto
- Tenor
- Baritone
- Bass

B Towards a Perceptually Plausibly Description of Expressive Tempo

This appendix describes an alternative method for computing expressive tempo, captured by the local beat period (BP), to the one described in performance codecs $\mathcal{C}_{1.0}$ and $\mathcal{C}_{2.0}$ defined in Sections 3.3.1 and 4.3.2, respectively. This alternative definition of the tempo conceptualizes expressive parameters in a way that is more aligned with the way humans perceive music.

As discussed in Sections 3.3.1 and 4.3.2, the BP can be computed by dividing the inter-onset interval (IOI) between two score positions by the nominal score duration of that interval (for instance, if a half-note interval between two onsets in a quarter-beat piece is found to correspond to an interval of 1s in a performance, the BP of the performance at that position would be 0.5s). However, there are several factors that complicate the computation of meaningful local tempo values from the note events in the performance.

First, this computation can lead to erratic jumps of consecutive BP values, especially when the IOIs are relatively small. Second, there is evidence that perceived beat times do not always coincide with the onset time of a note event at that beat: listeners prefer a beat grid that is slightly smoother than implied by the literal note onsets (Dixon et al., 2006).

A way to address these issues is not to use the performed note onset times directly to compute the BP values, but rather, use a cubic spline approximation of the performed note onset times, with a regularization term to obtain a slight smoothing effect. Additionally, we use a weighting scheme in the spline approximation that penalizes IOIs that are shorter than the median IOI of the piece, reducing the noise in BP values caused by small IOI values.

We can extend the idea described above to define a new version of the parameters capturing aspects of tempo, timing and articulation. In order to distinguish between the original definition of these parameters and the definition presented below, we will use *performance codec v. 1.5* ($\mathcal{C}_{1.5}$) to refer to a modified version of performance codec $\mathcal{C}_{1.0}$ that substitutes the tempo-related parameters (log BPR, timing and log articulation) with a more cognitively plausible version, as described below. Likewise, we use *performance codec v. 2.5* ($\mathcal{C}_{2.5}$) to refer to a modified version of performance codec $\mathcal{C}_{2.0}$ that substitutes the tempo-related parameters with a more cognitively plausible version. These cognitively plausible expressive parameters for tempo, timing and articulation are defined as follows:

- log BPR. We use the cubic spline interpolation of the onset times described above (which we denote as $\text{sinterp}(\cdot)$) to compute the estimated performed onset time $\hat{o}_i^{perf}$ of a score position o_i (instead of using the average onset time as in Section 3.3.1), as

$$\hat{o}_i^{perf} = \text{sinterp}(\text{onset}(o_i) \mid \mathbf{o}^{perf}, \boldsymbol{\mathcal{X}}) \tag{B.1}$$

  where $\mathbf{o}^{perf} = \{\text{onset}_{perf}(n_i) \mid n_i \in \boldsymbol{\mathcal{X}}\}$ is the set of performed onset times for each note in the score. The beat period for a note n_i is then given by

$$\text{BP}(n_i) = \frac{\text{IOI}^{perf}_{o(n_i)}}{\text{IOI}^{score}_{o(n_i)}}, \tag{B.2}$$

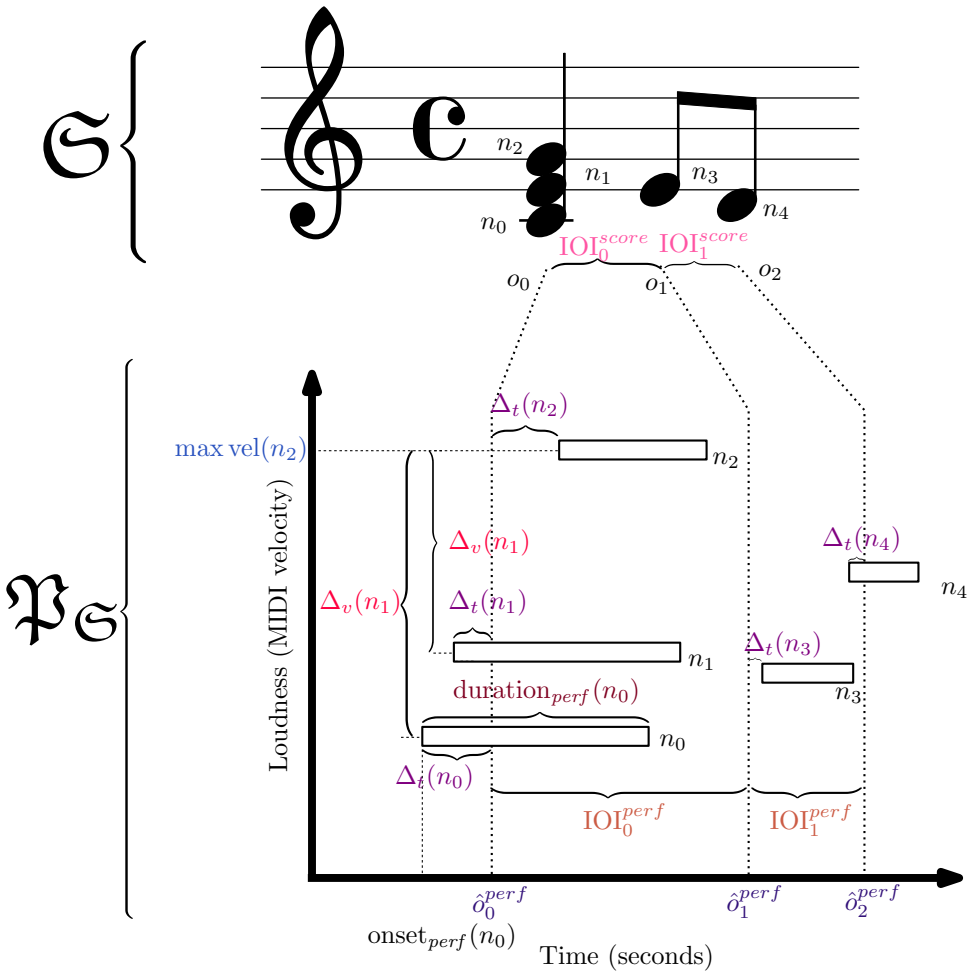


Figure B.1: Excerpt of a matched MIDI performance $\mathfrak{P}_{\mathfrak{S}}$ (as a piano-roll where the x axis denotes time and the y axis denotes MIDI velocity) and its corresponding score $\mathfrak{S}$ showcasing elements for computing the cognitively motivated expressive parameters relating to tempo, timing and articulation.

where $\text{IOI}^{perf}_{o(n_i)}$ and $\text{IOI}^{score}_{o(n_i)}$ are the (spline interpolated) performed and notated IOIs corresponding to note n_i, respectively, computed as

$$\text{IOI}_i^{perf} = \hat{o}_{i+1}^{perf} - \hat{o}_i^{perf} \qquad \text{and} \qquad \text{IOI}^{score} = \text{onset}(o_{i+1}) - \text{onset}(o_i) \tag{B.3}$$

Note that we choose to define parameter log BPR, such that the value of BP at the score onset o_i describes the local tempo between onsets o_i and o_{i+1} (rather than between o_{i-1} and o_i). This definition of the IOI is illustrated in Figure B.1 with IOI_0^{score} and IOI_1^{score} (in pink) representing score IOIs corresponding to score onsets o_0 and o_1, respectively. Finally, the log BPR is given by

$$\mathrm{y}_{\log \text{bpr}}(n_i) = \log_2 \frac{\text{BP}(n_i)}{\text{BP}_{ave}}, \tag{B.4}$$

where BP_{ave} is the average beat period.

Figure B.2 shows a comparison of the values of log BPR computed using the performance codec v. 1.0 defined in Section 3.3.1 and the smoothed version discussed here for the first 5 bars of Chopin's Nocturne Op. 9 No. 1 in B♭ minor. This plot showcases the issues with the previous version of the codec, which can result in artifacts for the tuples in bars 2 and 3.

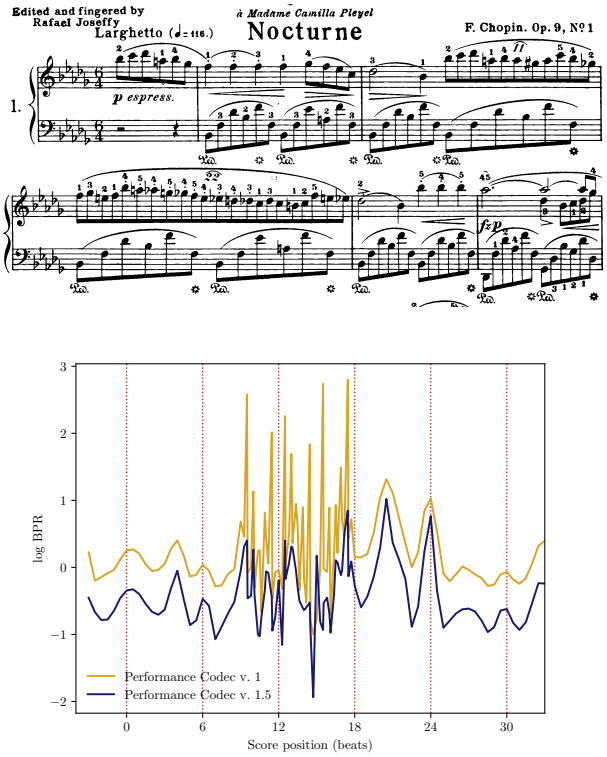


Figure B.2: Comparison of log BPR computed using performance codecs v. 1.0 and 1.5 for the first bars of Chopin's Nocturne Op. 9 No. 1 in B♭ minor performed by Nikita Magaloff. The red vertical dotted lines denote barlines. The excerpt from the score comes from Schirmer's Edition (Chopin, 1915a).

- Timing. This parameter is simply expressed as

$$\mathsf{y}_{\text{tim}}(n_i) = \hat{o}_i^{perf} - o^{perf}(n_i), \tag{B.5}$$

where $\hat{o}_i^{perf}$ is computed as in Equation (B.1). Figure B.1 illustrates this concept of timing for notes n_0, n_1 and n_2, n_3 and n_4 ($\Delta_t(n_0)$, $\Delta_t(n_1)$, $\Delta_t(n_2)$, $\Delta_t(n_3)$ and $\Delta_t(n_3)$ in violet). In contrast to the timing parameter in performance codecs $\mathcal{C}_{1.0}$ and $\mathcal{C}_{2.0}$, the above definition of beat period implies that the timing for score onsets consisting of a single note (as is the case of o_1 and o_2 in our running example) is no longer zero, as illustrated in Figure B.1 for n_3 and n_4 ($\hat{o}_1^{perf}$ and $\hat{o}_2^{perf}$ in indigo).

- log Articulation. We can compute this expressive parameter as

$$\mathsf{y}_{\log\text{art}}(n_i) = \log_2 \frac{\text{duration}_{perf}(n_i)}{\text{duration}(n_i)\text{BP}(n_i)}, \tag{B.6}$$

where $\text{BP}(n_i)$ is computed using Equation (B.2). By duration_{perf}, we denote the performed duration of a note, illustrated in Figure B.1 in burgundy for n_0.

C Modeling Joint Interactions of Expressive Parameters

C.1 Introduction

Through what is called *expressive interpretation*, the performance of a musical piece by a skilled musician can convey a variety of information to a listener, ranging from a particular mood to the elucidation of particular structural aspects of the music (voicing, phrasing, rhythm). In Western piano music of the common practice era, especially that of the Romantic period, expressive interpretation is characterized by a particularly free shaping of expressive parameters such as local tempo and dynamics, but also the timing and articulation of individual notes. A plausible assumption is that a performer uses such degrees of freedom jointly to realize a single expressive intention (even if this intention itself may be composed of several ideas), not least because a fully independent use of expressive parameters would imply a considerable cognitive burden on both the performer and the listener.

In this appendix we present an exploratory study on the joint interactions of expressive parameters, measured from recordings of classical piano performances by professional pianists. Our contribution consists of two parts. In the first part, we investigate the degree to which the measured data conveys monotonic, though possibly non-linear, pairwise interactions between expressive parameters (e.g. interactions between dynamics and tempo, timing vs articulation, etc.). Although we find some clear indications of these interactions, the results also show that the presence or absence of such interactions varies from piece to piece, and possibly from performer to performer.

In the second part, we report an experiment in which we use the non-linear basis function models (NBMs) described in Section 4.2 to test whether joint modeling of the expressive parameters improves the predictive accuracy of the model[1]. We do so by optimizing the accuracy of both joint and disjoint models for the expressive parameters in an extensive grid search over model hyperparameters. We find that for some of the expressive parameters, highest accuracy is achieved by joint models, whereas other expressive parameters can be better modeled individually.

The rest of this appendix is structured as follows. In Section C.2, we discuss evidence and theories for interactions between expressive parameters from the music psychological literature, as well as prior computational modeling approaches that involve joint modeling of several expressive parameters. Section C.3 provides an analysis of interactions between the expressive parameters, as measured from the data. The computational modeling setup and experiments are described in Section C.4. We discuss the results of both the empirical analysis and the modeling experiment in Section C.4.3, Finally, conclusions are presented in Section C.5.

[1]The experimental results reported in this appendix were conducted in early 2016, before the use of the RNN-based sequential models described in Section 4.3.

C.2 Evidence for Interactions from Music Psychology

Musicians' expressive manipulations of tempo, timing, dynamics, and articulation have been studied from a cognitive perspective, both individually and in combination, to determine how they shape listeners' perceptions of expressively performed music. A number of studies have sought to identify interactions between pairs of expressive parameters. In some such studies, listeners are presented with musical stimuli that vary on two dimensions (e.g. timing and dynamics), while all other parameters are held constant. The listeners are asked to judge one target dimension while ignoring the other. If the non-target dimension still affects listeners' responses, then the two parameters likely interact at some stage of perception, and this interaction cannot be avoided by channeling attention towards one parameter (Tekman, 2002).

Relationships have been observed between tempo and dynamics: for example, listeners perceive a faster tempo when melodies are played with a crescendo than when the same melodies are played with a decrescendo, even when the actual performed tempi are the same (Boltz, 2011). Timing and dynamics also interact in the context of accents, with interonset intervals perceived as longer when they precede loud notes than when they precede soft notes (Tekman, 2002). Tempo has been shown to relate to articulation as well; for instance, in one study, staccato passages were perceived as faster than legato passages (Geringer et al., 2006).

Other studies, rather than evaluating listeners' perceptions of specific expressive parameters, have investigated perceptions of musical expression more generally; for instance, by evaluating perceived emotion. The communication of emotion (including felt emotions that are induced by the music, and perceived emotions that are conveyed by the music but not felt by the listener) is recognized as a main component of musical expression (Juslin, 2003). Predictable combinations of expressive parameters have been shown to underlie the perception of specific emotional states. When listeners are asked to assign emotion labels to melodies, for example, the melodies they identify as 'sad' tend to be quiet and played with a legato articulation, while the melodies they identify as 'angry' tend to be loud and played with staccato articulation (Juslin and Laukka, 2003).

The relationships observed between particular parameter combinations and perceived emotions are not unique to music. In many cases, the same relationships are observed in speech (Juslin and Laukka, 2003). Parallels are also observed in nonverbal human behavior, suggesting that strategies for communicating emotion through music may have an origin in the way these emotions are communicated behaviorally (Eitan and Granot, 2006). Thus, interactions between expressive parameters in music could be partly attributable to the emotional qualities that a performer intends to communicate. To communicate a specific emotion, parameters must be jointly manipulated in particular ways.

Underlying listeners' perceptions of emotional expression in music is the build up, confirmation, and violation of their expectations (Ockelford, 2006). To an extent, listeners' expectations derive from fixed aspects of the musical structure (i.e. the tonal and temporal frameworks that constrain performers' expressive manipulations), but performers can also shape listeners' expectations by emphasizing or clarifying particular structural features.

Listener expectations are often investigated with tasks that require temporal synchronization of body movements with a sounded music performance. Listeners whose expectations align closely with a performer's intentions should be better able to synchronize their own movements with the sounded performance than would be the case for listeners who do not predict the performer's intentions as accurately (Keller et al., 2007). Research testing how precisely listeners synchronize their body movements (e.g. finger-taps, instrumental playing, or dance movements) with sounded music has shown that clear cues from performers indicating what they intend to play

next improves synchronization substantially (Goebl and Palmer, 2009). When performers can manipulate multiple expressive parameters simultaneously to emphasize a particular structural feature, their interpretation and intentions should become clearer, enabling listeners to construct more accurate expectations. It could be that performers who use predictable combinations of expressive parameters more consistently are easier to synchronize with and more successful at communicating expression; however, such a possibility has not yet been empirically tested.

While the music psychology literature provides some indication of how listeners expect pairs of expressive parameters to relate in certain (simplistic) contexts, it remains unclear whether these relationships are upheld during normal music performance, when the underlying piece structure is complex and many expressive parameters must be manipulated in parallel. We do know that in some cases, performers will manipulate two parameters in different ways during the course of a single piece to achieve different expressive goals (e.g. slowing down while simultaneously getting softer, then elsewhere slowing down while getting louder). Whether the consistent use of particular parameter relationships relates to the aesthetic quality of a performance, increases its predictability, or makes the communication of expression more successful likewise requires further study.

C.3 Empirical Analysis of Recorded Piano Performances

In this section, we present some empirical evidence for (pairwise) interactions between expressive parameters, as extracted from the Magaloff/Chopin and Zeilinger/Beethoven datasets using performance codec $\mathcal{C}_{1.5}$ defined in Appendix B. This performance codec consists of four expressive parameters that describe aspects of a performance in a note-wise fashion (i.e. there is a value of the expressive parameters for each note in the score). The parameters in this performance codec are MIDI velocity, which captures aspects of dynamics; $\log \mathrm{BPR}$, which captures local tempo; timing, which captures the temporal shifting of the individual notes from the implied metrical grid defined by the local tempo; and $\log$ articulation, which captures the articulation of individual notes. For a more detailed description of these parameters see Section 3.3.1 and Appendix B.

Figure C.1 shows a boxplot of Spearman's ρ rank correlation between expressive parameters for all performed pieces by each pianist. For each piece in a dataset, we estimate Spearman's ρ for each pairwise combination of expressive parameters (e.g. MIDI velocity and $\log \mathrm{BPR}$, MIDI velocity and timing, MIDI velocity and $\log$ articulation, etc.). Spearman's ρ provides some evidence of the inter-interactions between the expressive parameters, by measuring how well the relationship between two expressive parameters can be described using a (non-linear) monotonic function. Here it is possible to see that the pairwise correlations tend to be stronger and more consistent for the Zeilinger/Beethoven dataset, with larger absolute values for the mean values and smaller variances. From a musical perspective this is somewhat expected, since the Magaloff/Chopin corpus contains pieces from a wider variety of musical forms (preludes, waltzes, etudes, etc.) while the Zeilinger/Beethoven corpus is restricted to piano sonatas. Nevertheless, both pianists follow similar trends, with expressive dynamics tending to be negatively correlated with BPRs, but positively correlated with articulation and timing. In Zeilinger's case, there is a rather weak correlation between BPR and timing, which for Magaloff is effectively zero, and both present a similar correlation between timing and articulation. These results suggest the existence of general principles of expressive performance that are independent of the pianist or of the style.

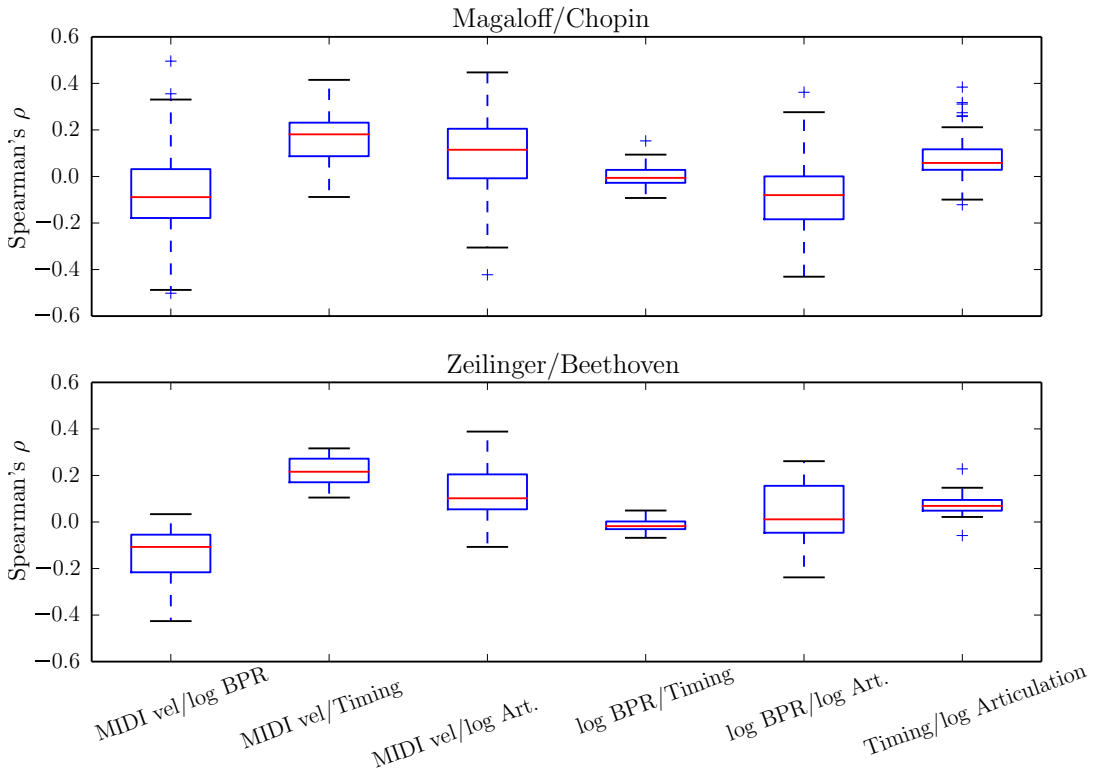


Figure C.1: Spearman's ρ between expressive targets, measured from the Magaloff/Chopin corpus (top), and the Zeilinger/Beethoven corpus (bottom)

C.4 Joint Modeling of Expressive Parameters

In this section we describe a preliminary study that aimed to determine whether computational models of expression benefit from explicitly modeling expressive parameters jointly.

The BMs described in Chapters 3 and 4 focus on modeling expressive parameters independently of each other. Nevertheless, evidence from the music psychology literature, as well as the empirical analysis of the performance data described above in Section C.3, suggest that there performers shape expressive performances through deliberate interactions between expressive parameters.

In this section we describe a series of quantitative experiments that aimed to determine whether modeling expressive parameters jointly or in an independent fashion leads to better predictions. More specifically, we conducted a series of cross-validation experiments that compared the predictive accuracy of NBMs which model parameters independently to that of NBMs which model parameters jointly.

The rest of this section is structured as follows: Section C.4.1 describes how to adapt the NBMs to jointly model expressive parameters. In Section C.4.2 we describe how the cross-validation experiments were conducted. Section C.4.3 discusses the results of this evaluation.

C.4.1 Joint Modeling of Expressive Parameters Using Non-linear Basis Models

In this section, we describe a simple extension of the NBMs described in Section 4.2.2 to allow for modeling expressive parameters jointly. As discussed before, NBMs are models that use FFNNs to model the relationship between score properties (captured in terms of the basis functions) and the expressive parameters.

Instead of having an FFNN modeling each expressive parameter, we can define a single FFNN describing all (or a combination of) parameters. The main difference is that instead of having an output layer with a single unit, the output layer of an FFNN modeling parameters jointly has a unit for each expressive parameter. We can write the output of such a network (with L layers) which models N_y parameters for score note n_i as

$$f_{nbm}^{(joint)}(n_i) = \sigma^{(L)}\left(\mathbf{w}^{(L)}\boldsymbol{h}_i^{(L-1)} + \mathbf{w}_0^{(L)}\right), \tag{C.1}$$

where $\boldsymbol{h}_i^{(L-1)} \in \mathbb{R}^{D_{L-1}}$ is the vector of (hidden) activations of the $L-1$ layer for score note n_i, $\mathbf{w}^{(L)} \in \mathbb{R}^{N_y \times D_{L-1}}$ is a matrix of weights connecting the hidden activations of the $L-1$ layer with layer L, $\mathbf{w}_0^{(L)} \in \mathbb{R}^{N_y}$ is a bias vector, and $\sigma^{(L)}$ is an element-wise activation function. The rest of the network has the same form as discussed in Section 4.2.2. Since we treat the modeling of expressive parameters as a regression problem, we use a linear activation for the output layer (i.e. $\sigma^{(L)}(\xi) = \xi$).

C.4.2 Experiments

In this section, we describe the cross-validation experiments conducted to assess whether NBMs benefit from modeling expressive parameters jointly. For these experiments, we use the Magaloff/Chopin and the Zeilinger/Beethoven datasets. As in the case of the empirical analysis presented in Section C.3, we describe the performances in terms of the four expressive parameters defined in performance codec $\mathcal{C}_{1.5}$, namely MIDI velocity, log BPR, timing and log articulation. In this section, we will refer to a joint model as a model that predicts all expressive parameters jointly, and to an independent model as a model that predicts a single expressive parameter.

Several 5-fold cross-validations were conducted for each modeling approach (joint and independent), dataset (Magaloff/Chopin and Zeilinger/Beethoven) and FFNN architecture combination (21 architectures, see description below). The score information in the datasets was represented using basis functions from the groups described in Sections 3.3.2 and 4.2.1. The cross-validations were conducted as follows: We defined training and test sets randomly for each of the 5 partitions for each dataset, such that each piece in a dataset occurred exactly once per test set. The training set for each partition contained 80% of the pieces in the dataset, and the test set contained the remaining 20%. For each of these partitions, we trained joint and independent models for each architecture on the training set and then used the trained models to predict the expressive parameters for each piece in the test set. We use the coefficient of determination (R^2) and Pearson's correlation coefficient (r) to characterize the accuracy of the learned models. R^2 is a measure of the variance explained by the model, when a linear relationship between the predictions and the targets exists. Note that R^2 can be negative in cases where the model residual has a larger variance than the signal itself, implying a poor fit of the model to the data. We rank our results according to R^2, since it is a more robust measure of the performance of a model.

As mentioned above, we performed an extensive grid search over the network architecture, to try to find the optimal settings which maximized the predictive accuracy for each joint and

independent model. This search considered FFNNs with up to three hidden layers of different sizes. All configurations were used for the modeling individual parameters, as well as for joint modeling. For the first layer, the possible sizes were 20, 100 and 500 hidden units; while two-layered models had an additional 5, 20 or 100 units in the second layer, and three layered models had an additional 5 or 20 units in the third layer. All of the hidden layers had rectified linear units, and the output layer had linear units. In the following discussion, we will denote these architectures as (D_1, D_2, D_3), where D_1, D_2 and D_3 are the number of units in the first, second and third hidden layers, respectively.

All models were trained by minimizing the squared error between the predictions and the expressive parameters using RMSProp, using a learning rate of 10^{-5} and a gradient decay parameter of 0.9. We used 80% of the pieces in the training set for updating the parameters and 20% as a validation set. In order to avoid overfitting, we used dropout (with $p_{dropout} = 0.5$), l_2-norm weight regularization (with regularization coefficient of 10^{-3}) and early stopping after 100 epochs without improvement in the loss of the validation set. The weights of the layers are randomly initialized as proposed by Glorot and Bengio (2010), and the biases were initialized to zero. All models were trained for a maximum of 2000 epochs.

C.4.3 Results and Discussion

The results of the grid search of the 5-fold cross-validations are presented in Tables C.1, C.2, C.3 and C.4 for MIDI velocity, log BPR, timing and log articulation, respectively. In these tables, the results correspond to the R^2 and r averaged across all pieces in the test set. In these tables, the R^2 values in boldface in the columns "Independent" and "Joint" for each dataset correspond to the best results (the highest R^2) for each expressive parameter.

We are interested in whether modeling parameters independently versus jointly has an effect on predictive accuracy. For this purpose we made two kinds of comparisons: First, we tested for an overall effect across network architectures of modeling parameters independently or jointly; second, we compared the best architectures achieved from modeling parameters independently with the best architectures achieved from modeling parameters jointly.

For the first comparison, we conducted an independent samples two-tailed t-test at the $p < 0.01$ level to evaluate the difference in R^2 between all architectures modeling expressive parameters independently and all architectures modeling expressive parameters jointly, for each expressive parameter/dataset combination (for a total of 4 parameters $\times$ 2 datasets = 8 tests). The results of these comparisons are summarized as follows:

- For MIDI velocity and log articulation, we found no significant differences in R^2 for either the Magaloff/Chopin or Zeilinger/Beethoven datasets.
- For log BPR in architectures trained on the Magaloff/Chopin dataset, R^2 was significantly *higher* for architectures modeling parameters *independently* (mean = 0.012, std = 0.005) than for architectures modeling parameters jointly (mean = 0.006, std = 0.007, t(40) = 2.83, $p = 0.007$, Cohen's $d = 0.87$). On the other hand, no significant difference was found for the Zeilinger/Beethoven dataset.
- For timing, we found no significant difference in R^2 for the Magaloff/Chopin dataset. On the other hand, for architectures trained on the Zeilinger/Beethoven dataset, R^2 was significantly *higher* for architectures modeling parameters *jointly* (mean = 0.003, std = 0.003) than for architectures modeling parameters independently (mean < 0.001, std = 0.005, t(38) = 3.93, $p < 0.001$, Cohen's $d = 1.24$).

	Magaloff/Chopin				**Zeilinger/Beethoven**			
	Independent		Joint		Independent		Joint	
Architecture	R^2	r	R^2	r	R^2	r	R^2	r
(20, 0, 0)	0.167	0.458	0.168	0.453	0.194	0.543	0.193	0.519
(20, 5, 0)	0.167	0.458	0.161	0.460	0.165	0.438	0.130	0.464
(20, 5, 5)	0.141	0.457	0.124	0.444	0.139	0.516	0.016	0.135
(20, 5, 20)	0.155	0.445	0.113	0.431	0.105	0.303	0.033	0.111
(20, 20, 0)	0.168	0.451	0.167	0.456	0.193	0.526	0.189	0.514
(20, 20, 5)	0.002	0.074	0.083	0.331	0.000	0.054	0.025	0.186
(20, 20, 20)	0.157	0.435	0.150	0.428	0.075	0.213	0.152	0.475
(20, 100, 0)	0.161	0.448	0.165	0.451	0.192	0.518	0.200	0.523
(20, 100, 5)	0.141	0.455	0.149	0.443	0.004	0.097	0.022	0.090
(20, 100, 20)	0.165	0.446	0.150	0.442	0.196	0.517	0.185	0.515
(100, 0, 0)	0.164	0.457	0.165	0.449	0.201	0.550	0.203	0.533
(100, 5, 0)	0.152	0.417	0.155	0.441	0.178	0.479	0.075	0.282
(100, 5, 5)	0.162	0.453	0.053	0.363	0.145	0.421	0.077	0.357
(100, 5, 20)	0.034	0.116	0.071	0.271	0.202	0.539	0.124	0.397
(100, 20, 0)	0.174	0.460	**0.179**	0.463	0.211	0.546	0.222	0.545
(100, 100, 0)	0.168	0.457	0.174	0.460	0.195	0.538	0.205	0.528
(100, 100, 5)	0.176	0.458	0.151	0.415	0.126	0.367	-0.003	-0.017
(100, 100, 20)	**0.176**	0.459	0.170	0.452	0.208	0.541	0.117	0.347
(500, 0, 0)	0.163	0.459	0.166	0.459	0.207	0.556	0.212	0.546
(500, 5, 5)	0.138	0.418	0.110	0.407	0.185	0.537	0.096	0.344
(500, 100, 20)	0.175	0.456	0.177	0.457	**0.226**	0.554	**0.228**	0.546

Table C.1: Predictive results for MIDI velocity for each architecture, averaged over a 5-fold cross-validation on the Magaloff/Chopin and Zeilinger/Beethoven piano performance corpora. A larger R^2 and r means better performance. The numbers in bold represent the best architecture (in terms of R^2) for each condition (Independent or Joint). See text for description of the neural architectures.

For the second comparison, we conducted a paired-samples two-tailed t-test at the $p < 0.01$ level to evaluate the differences in R^2 between the best architectures modeling parameters independently and the best architectures modeling parameters jointly, for each expressive parameter/dataset combination (for a total of 4 parameters $\times$ 2 datasets = 8 tests). According to these t-tests, none of the differences was statistically significant.

The results of these experiments are inconclusive: there seems to be no modeling strategy that consistently leads to better predictions. Nevertheless, these negative results do not necessarily imply there is no benefit in modeling parameters jointly: A possible reason for these results might be that the FFNNs described in Section C.4.1 do not allow for explicitly modeling the interactions between expressive parameters (e.g. explicitly capturing the correlations between expressive parameters). Furthermore, these models do not take multiple expressive parameters as input when predicting a given expressive parameter (e.g. do not consider the log BPR, timing or log articulation of a note for predicting the MIDI velocity of that note). Given the complex nature of the interactions between expressive parameters, models that explicitly account for these interactions may prove more successful than models that consider individual parameters in isolation.

	Magaloff/Chopin				Zeilinger/Beethoven			
	Independent		Joint		Independent		Joint	
Architecture	R^2	r	R^2	r	R^2	r	R^2	r
(20, 0, 0)	0.016	0.137	**0.019**	0.167	0.010	0.151	0.007	0.204
(20, 5, 0)	0.006	0.072	0.003	0.071	0.016	0.132	0.002	0.050
(20, 5, 5)	0.007	0.082	0.000	0.028	0.004	0.061	-0.002	-0.040
(20, 5, 20)	0.004	0.068	0.001	0.046	0.004	0.034	0.001	0.044
(20, 20, 0)	**0.023**	0.162	0.012	0.133	0.000	0.106	0.000	0.169
(20, 20, 5)	0.010	0.097	0.000	0.011	0.011	0.102	-0.002	0.010
(20, 20, 20)	0.011	0.108	0.000	0.033	0.006	0.101	0.012	0.167
(20, 100, 0)	0.016	0.134	0.017	0.156	0.013	0.169	0.006	0.192
(20, 100, 5)	0.005	0.064	0.000	0.022	0.005	0.078	0.000	0.030
(20, 100, 20)	0.007	0.088	0.004	0.075	0.014	0.147	0.016	0.180
(100, 0, 0)	0.018	0.161	0.015	0.151	0.006	0.143	0.003	0.165
(100, 5, 0)	0.009	0.112	0.003	0.073	0.012	0.161	0.002	0.084
(100, 5, 5)	0.017	0.155	0.000	0.031	0.007	0.079	-0.001	-0.008
(100, 5, 20)	0.004	0.052	0.000	0.013	0.001	0.021	0.006	0.113
(100, 20, 0)	0.017	0.145	0.016	0.148	0.021	0.177	0.008	0.192
(100, 100, 0)	0.015	0.141	0.015	0.145	0.005	0.124	0.008	0.200
(100, 100, 5)	0.017	0.151	-0.001	0.001	0.018	0.185	0.002	0.050
(100, 100, 20)	0.017	0.150	0.010	0.107	0.010	0.119	0.004	0.093
(500, 0, 0)	0.011	0.140	0.014	0.164	-0.004	0.118	0.001	0.206
(500, 5, 5)	0.011	0.119	0.002	0.056	**0.025**	0.176	0.012	0.115
(500, 100, 20)	0.014	0.134	0.012	0.124	0.006	0.154	**0.016**	0.199

Table C.2: Predictive results for log BPR for each architecture, averaged over a 5-fold cross-validation on the Magaloff/Chopin and Zeilinger/Beethoven piano performance corpora. A larger R^2 and r means better performance. The numbers in bold represent the best architecture (in terms of R^2) for each condition (Independent or Joint). See text for description of the neural architectures.

C.5 Conclusions

In this appendix we presented preliminary work studying the joint interactions of dimensions describing aspects of expressive performances. The empirical analysis of piano performances described in Section C.3 shows some evidence supporting the intuition that different aspects of musical expression are not independent from each other. This has also been confirmed by the music psychology literature.

In this appendix we conducted experiments using NBMs to compare joint and independent modeling of expressive parameters on two corpora of performances of piano music from the classical and romantic periods. Analysis of the performances show some evidence supporting the intuition that different aspects of musical expression are not independent from each other. Nevertheless, the results of the cross-validation experiments testing whether modeling parameters independently or jointly led to better predictive models are inconclusive: it is not clear if the models would benefit from joint modeling. Future work will include an in-depth quantitative evaluation of the predictive accuracy of models describing expressive parameters jointly vs. models describing parameters independently.

	Magaloff/Chopin				**Zeilinger/Beethoven**			
	Independent		Joint		Independent		Joint	
Architecture	R^2	r	R^2	r	R^2	r	R^2	r
(20, 0, 0)	0.007	0.077	**0.010**	0.104	-0.002	0.054	**0.008**	0.105
(20, 5, 0)	0.008	0.094	0.005	0.084	-0.002	0.007	0.000	0.026
(20, 5, 5)	0.008	0.106	0.004	0.080	0.000	0.006	0.000	0.027
(20, 5, 20)	0.007	0.090	0.003	0.077	-0.001	0.007	0.001	0.028
(20, 20, 0)	0.005	0.072	0.006	0.089	-0.001	0.038	0.004	0.080
(20, 20, 5)	0.008	0.099	0.002	0.064	0.000	0.004	-0.002	0.022
(20, 20, 20)	0.005	0.077	0.005	0.078	0.001	0.043	0.002	0.055
(20, 100, 0)	0.006	0.080	0.007	0.090	0.000	0.029	0.003	0.083
(20, 100, 5)	0.009	0.103	0.004	0.075	0.001	0.035	0.000	0.020
(20, 100, 20)	0.003	0.063	0.005	0.083	-0.002	0.054	0.004	0.080
(100, 0, 0)	0.016	0.133	0.008	0.089	-0.002	0.063	0.008	0.109
(100, 5, 0)	0.005	0.068	0.006	0.090	0.001	0.054	0.003	0.051
(100, 5, 5)	0.008	0.105	0.002	0.072	0.000	0.037	0.001	0.017
(100, 5, 20)	0.010	0.116	0.002	0.046	0.000	0.010	0.003	0.063
(100, 20, 0)	**0.011**	0.107	0.008	0.098	-0.002	0.034	0.009	0.110
(100, 100, 0)	0.004	0.090	0.009	0.097	0.000	0.056	0.006	0.097
(100, 100, 5)	0.008	0.093	0.005	0.084	0.000	0.027	-0.002	0.011
(100, 100, 20)	0.008	0.099	0.006	0.085	0.002	0.078	0.003	0.052
(500, 0, 0)	0.006	0.098	0.010	0.102	-0.001	0.071	0.007	0.112
(500, 5, 5)	0.004	0.067	0.004	0.078	**0.003**	0.062	0.005	0.071
(500, 100, 20)	0.006	0.100	0.007	0.095	0.002	0.053	0.008	0.100

Table C.3: Predictive results for timing for each architecture, averaged over a 5-fold cross-validation on the Magaloff/Chopin and Zeilinger/Beethoven piano performance corpora. A larger R^2 and r means better performance. The numbers in bold represent the best architecture (in terms of R^2) for each condition (Independent or Joint). See text for description of the neural architectures.

	Magaloff/Chopin				**Zeilinger/Beethoven**			
	Independent		Joint		Independent		Joint	
Architecture	R^2	r	R^2	r	R^2	r	R^2	r
(20, 0, 0)	0.065	0.278	0.073	0.284	0.065	0.274	0.089	0.315
(20, 5, 0)	0.068	0.284	0.042	0.234	0.065	0.211	0.045	0.263
(20, 5, 5)	0.043	0.216	0.023	0.189	0.048	0.244	0.001	0.032
(20, 5, 20)	0.051	0.233	0.02	0.178	0.023	0.112	0.007	0.072
(20, 20, 0)	0.069	0.282	0.066	0.279	0.051	0.216	0.087	0.323
(20, 20, 5)	0.012	0.061	0.015	0.132	0.017	0.096	0.002	0.042
(20, 20, 20)	0.063	0.276	0.029	0.188	0.001	0.046	0.028	0.224
(20, 100, 0)	0.067	0.279	0.066	0.278	0.065	0.264	0.085	0.311
(20, 100, 5)	0.063	0.287	0.025	0.179	0.001	0.049	0.004	0.023
(20, 100, 20)	0.062	0.278	0.044	0.24	0.075	0.268	0.057	0.271
(100, 0, 0)	0.067	0.284	0.068	0.279	0.080	0.298	0.074	0.269
(100, 5, 0)	0.058	0.257	0.046	0.249	0.102	0.331	0.028	0.138
(100, 5, 5)	0.064	0.279	0.017	0.207	0.078	0.28	0.016	0.167
(100, 5, 20)	0.010	0.081	0.012	0.104	0.080	0.300	0.034	0.223
(100, 20, 0)	0.070	0.280	0.071	0.28	0.079	0.306	0.105	0.34
(100, 100, 0)	0.068	0.282	**0.075**	0.289	0.082	0.294	**0.107**	0.338
(100, 100, 5)	0.068	0.281	0.033	0.196	0.024	0.113	0.002	0.06
(100, 100, 20)	0.068	0.281	0.066	0.276	0.065	0.227	0.061	0.226
(500, 0, 0)	0.064	0.281	0.072	0.292	0.073	0.29	0.104	0.353
(500, 5, 5)	0.059	0.264	0.027	0.194	**0.100**	0.367	0.019	0.182
(500, 100, 20)	**0.070**	0.286	0.068	0.278	0.087	0.326	0.101	0.343

Table C.4: Predictive results for log articulation for each architecture, averaged over a 5-fold cross-validation on the Magaloff/Chopin and Zeilinger/Beethoven piano performance corpora. A larger R^2 and r means better performance. The numbers in bold represent the best architecture (in terms of R^2) for each condition (Independent or Joint). See text for description of the neural architectures.

D Probabilistic Non-linear Basis Models Using Gaussian Mixture Density Networks

D.1 Introduction

The linear and non-linear variants of the BMs described in the main text model only a single *global* performance strategy: the models can only represent a performance that is the "average" of all performance strategies. For example, the "average" performance of such a model might not be good enough to reproduce the specific timing patterns of a waltz from a mazurka (both of which are in $\frac{3}{4}$). A musically plausible assumption is that instead of a single *global* performance strategy there are many *local* performance strategies. A performer then shapes an expressive performance by moving (i.e. selecting) among these performance strategies. From a probabilistic perspective, we can think of the approach of modeling global performance strategies as describing performances through unimodal distributions. On the other hand, modeling the local performance strategies can be embedded into the semantics of multimodal distributions, where each of the modes correspond to the characteristics of a (local) performance strategy.

In this appendix we present a probabilistic extension to the neural networks discussed in Chapter 4, using the framework of Mixture Density Networks (MDNs) (Bishop, 1994). This framework combines the descriptive power of *mixture models*, which are probabilistic models that allow for representing multimodal data through a (generally additive) mixture of probability distributions, with the approximation power and flexibility of *neural networks*. In a nutshell, an MDN is a neural network whose outputs parametrize a multimodal conditional probability distribution.

Below, we discuss how to use Gaussian MDNs (GMDNs) to parametrize the conditional probability distribution of the expressive parameters given the input score information, represented through basis functions. As their name suggests, GMDNs are neural networks that parametrize Gaussian mixture models (GMMs), i.e. that parametrize a mixture of Gaussian distributions. The main motivation for using these networks is to address two issues with the predictive models described in the main text:

1. GMDNs relax the restriction of the predictions of the models being unimodal (explicitly made by the linear models described in Chapter 3 and implicitly made by the neural networks described in Chapters 4); and

2. if used to jointly model expressive parameters, GMDNs describe the correlation between expressive parameters (see discussion in Appendix C.4.3).

In this appendix we present a mathematical description of GMDNs, as well as an experimental evaluation focusing on the first issue described above, namely, exploring whether the predictive accuracy of the models benefits from explicitly describing expressive parameters using multimodal distributions.

Following the notation established in Chapters 3 and 4 (in particular, Sections 3.2, 4.2 and 4.3), elements of a score $\mathfrak{S}$ are denoted as x (these elements can be single notes or onsets, denoted

as n and o, respectively); $\mathcal{Y}(x_i) = \boldsymbol{t}_i \in \mathbb{R}^{N_y}$ is a vector whose components represent expressive parameters for score element x_i; and $\mathcal{F}(x_i) = \boldsymbol{\varphi}(x_i) \in \mathbb{R}^M$ is vector whose components are the value of the basis functions representing score element x_i.

The rest of this appendix is structured as follows: Section D.2 presents a mathematical description of GMDNs. Section D.3 describes an experimental evaluation of GMDNs for predicting expressive dynamics. Section D.4 discusses the results of the experimental evaluation. Section D.5 discusses how to use GMDNs to visualize interactions between expressive parameters. Finally, Section D.6 concludes the appendix.

D.2 Model Description

In this section, we provide a mathematical description of GMDNs. In this discussion, we focus on the more general case of GMDNs modeling expressive parameters jointly (i.e. where the MDNs parametrize a mixture of multivariate Gaussian distributions), since the univariate case can be derived from the multivariate case in a straightforward way. We can use GMDNs to describe the conditional probability distribution of the expressive parameters $\boldsymbol{t}_i$ given the input basis functions $\boldsymbol{\varphi}(x_i)$ as

$$p(\boldsymbol{t}_i \mid \boldsymbol{\varphi}(x_i)) = \sum_{k=1}^{K} \rho_k\left(\boldsymbol{\varphi}(x_i)\right) \mathcal{N}\left(\boldsymbol{t}_i \mid \boldsymbol{\mu}_k\left(\boldsymbol{\varphi}(x_i)\right), \boldsymbol{\Sigma}_k\left(\boldsymbol{\varphi}(x_i)\right)\right) \tag{D.1}$$

where K is the number of components of the mixture; and $\rho_k\left(\boldsymbol{\varphi}(x_i)\right)$, $\boldsymbol{\mu}_k\left(\boldsymbol{\varphi}(x_i)\right)$ and $\boldsymbol{\Sigma}_k\left(\boldsymbol{\varphi}(x_i)\right)$ are the mixture coefficient, the mean and covariance matrix of the k-th component for score element x_i, respectively and $\mathcal{N}(\cdot \mid \boldsymbol{\mu}, \boldsymbol{\Sigma})$ represents the multivariate Gaussian distribution. In this case, the output of the neural network for $\boldsymbol{\varphi}(x_i)$, denoted as $\boldsymbol{y}_i \in \mathbb{R}^{\frac{K}{2}\left(2+3N_y+N_y^2\right)}$ has K units determining the mixture coefficients, KN_y units determining the means and $KN_y\frac{(N_y+1)}{2}$ determining the elements of the covariances matrices[1] for a total of $\frac{K}{2}\left(2+3N_y+N_y^2\right)$ units. To unclutter notation, in the following discussion we will write $\boldsymbol{\mu}_{ik} = \boldsymbol{\mu}_k\left(\boldsymbol{\varphi}(x_i)\right)$ and $\boldsymbol{\Sigma}_{ik} = \boldsymbol{\Sigma}_k\left(\boldsymbol{\varphi}(x_i)\right)$.

The mixture coefficients and means are given using the following parametrization of the output of the neural network

$$\rho_k\left(\boldsymbol{\varphi}(x_i)\right) = \frac{\exp\left(y^{\rho}_{k,i}\right)}{\sum_{j=1}^{K}\exp\left(y^{\rho}_{j,i}\right)} \tag{D.2}$$

$$\mu_{kq}\left(\boldsymbol{\varphi}(x_i)\right) = y^{\mu}_{kq,i}, \tag{D.3}$$

where $y^{\rho}_{q,i}$ represents the activation of the q-th unit of the subset of units determining the mixture coefficients for the i-th score element x_i, $\mu_{kq}\left(\boldsymbol{\varphi}(x_i)\right)$ is the q-th element of the mean of the k-th component at the i-th score element and $y^{\mu}_{kq,i}$ is the activation of the q-th unit corresponding to the subset of units determining the k-th mean.

Parametrizing the covariance matrices is trickier, since covariance matrices must be positive (semi) definite. Following (Williams, 1996), we use the Cholesky factorization of the inverse of the covariance matrix of the k-th component as

$$\boldsymbol{\Sigma}_k\left(\boldsymbol{\varphi}(x_i)\right)^{-1} = \mathbf{U}_{ki}^{T}\mathbf{U}_{ki}, \tag{D.4}$$

[1] Since covariance matrices are symmetrical instead of N_y^2 different elements, we only need N_y elements for representing the diagonal and $\frac{N_y(N_y-1)}{2}$ for representing the non-diagonal elements.

where $\mathbf{U}_{ki} \in \mathbb{R}^{N_y \times N_y}$ is an upper triangular matrix. The elements in the diagonal of this matrix are denoted $u_{kqq,i}$. These elements must be positive to ensure that the covariance matrix is positive semidefinite. The non-zero non-diagonal elements are denoted by $u_{kqj,i}$. These elements can be expressed in terms of the outputs of the network as follows:

$$u_{kqq,i} = \exp\left(y_{kqq,i}^{\Sigma_d}\right) \tag{D.5}$$

$$u_{kqj,i} = y_{kqj,i}^{\Sigma_{nd}}, \tag{D.6}$$

where $y_{kqq,t}^{\Sigma_d}$ and $y_{kqj,t}^{\Sigma_{nd}}$ are the activations of the subset of units determining the qq (diagonal) and qj (non-zero non-diagonal) elements of the upper triangular matrix resulting of the Cholesky factorization of the inverse covariance matrix of the k-th component, respectively. It is important to note that by using this parametrization we *do not have* to compute the Cholesky factorization of any covariance matrix, as the model *learns* the factorization by itself. In the case of a univariate distribution, the covariance is simply a scalar, which is given by Equation (D.5) (i.e. there are no non-diagonal elements).

A GMDN provides a joint probability distribution over the values of the expressive parameters instead of an estimate of these parameters. Therefore, to define predictive functions $\boldsymbol{f}_{(r)nbm_{gmd}}$, we require a method for generating estimates from the conditional probability distribution[2]. Two ways to generate such an estimate are:

1. Taking the expected value of the distribution, i.e.,

$$\boldsymbol{f}_{(r)nbm_{gmd}}(x_i) = \mathbb{E}\left\{p(\boldsymbol{t}_i \mid \boldsymbol{\varphi}(x_i))\right\} = \sum_{k=1}^{K} \rho_k\left(\boldsymbol{\varphi}(x_i)\right) \cdot \boldsymbol{\mu}_k\left(\boldsymbol{\varphi}(x_i)\right) \tag{D.7}$$

2. Taking the mean value of the largest component, i.e.,

$$\boldsymbol{f}_{(r)nbm_{gmd}}(x_i) = \boldsymbol{\mu}_{k^*}\left(\boldsymbol{\varphi}(x_i)\right), \tag{D.8}$$

   were $k^* = \operatorname{argmax}_k \rho_k(\boldsymbol{\varphi}(x_i))$ is the component with the largest mixing coefficient.

Training

Given a training set $\boldsymbol{\mathcal{T}}$, a GMDN can be trained by maximizing the log-likelihood given by

$$\begin{aligned}\mathcal{L}_{ll}(\boldsymbol{\theta}) &= \log p(\mathbf{t}_{\boldsymbol{\mathcal{T}}} \mid \boldsymbol{\Phi}_{\boldsymbol{\mathcal{T}}}, \boldsymbol{\theta}) \\ &= \sum_{\boldsymbol{t}_i \in \boldsymbol{\mathcal{T}}} \log\left\{\sum_{k=1}^{K} \rho_{tk} \mathcal{N}(\boldsymbol{t}_i \mid \boldsymbol{\mu}_{ik}, \boldsymbol{\Sigma}_{ik})\right\}\end{aligned} \tag{D.9}$$

which can be done using stochastic gradient descent (SGD).

D.3 Experiments

In this section, we describe a quantitative evaluation of the predictive accuracy of GMDNs to predict expressive dynamics, as captured by the (normalized) MIDI velocity of each note,

[2]Note that the formalism of GMDNs can be used for both FFNNs and RNNs, and therefore we write a single predictive function, meant to represent both cases, which is denoted with the $(r)nbm$ subscript.

described in the performance codec $\mathcal{C}_{1.0}$ defined in Section 3.3.1 (see Equation (3.3)). Since performance codec $\mathcal{C}_{1.0}$ describes expressive parameters in a note-wise fashion, we used non-linear BMs (NBMs) where the predictive function is an FFNN-based GMDNs, which we will refer to as GMDFFNNs, to model the conditional probability distribution of the MIDI velocity of each note given the representation of that note through basis functions.

The main objective of this evaluation was to determine whether using more than one component improves the predictive accuracy of the models. As a secondary objective, we investigated which of two strategies for generating predictions of the MDNs worked best – taking the expected value or the mean of the maximal component of the mixture.

We performed six 5-fold cross-validations using the Magaloff/Chopin and Zeilinger/Beethoven datasets for GMDFFNNs with 1, 2 and 5 components (2 datasets $\times$ 3 GMDFFNN architectures (with 1, 2 or 5 components) = 6 cross-validations). The score information in the datasets was represented using basis functions from the groups described in Sections 3.3.2 and 4.2.1. Training and tests sets were randomly defined for each of the 5 partitions for each dataset, so that each piece in a dataset occurred exactly once per test set. The training set for each partition contained 80% of the pieces in the dataset, and the test set contained the remaining 20%. For each partition, we trained a GMDFFNN for maximizing the log-likelihood of the observed MIDI velocity of each note given the representation through basis functions of that note. As discussed above, MDNs do not directly provide predictions of the MIDI velocity, instead, we generated model predictions using the two methods described above in Section D.2 (i.e. by taking the expected value, which we denote in the following discussion as "Expected Value", or the mean of the maximal component of the mixture, which we denote as "Max. Component"). This means that each trained GMDFFNN generates two sets of predictions, We use the coefficient of determination (R^2) and Pearson's correlation coefficient (r) to characterize the accuracy of the learned models.

D.3.1 Model Architectures and Training

The architecture of each GMDN consisted of two hidden layers with 100 and 20 rectified linear units each, and a output layer parametrizing a GMM with 1, 2 and 5 components, respectively.

These networks were trained using RMSProp, a variant of SGD where the step size is adaptively updated by a running average of the magnitude of the gradient, using a learning rate of 10^{-5} and a gradient decay parameter of 0.9. We used 80% of the pieces in the training set for updating the parameters and 20% as a validation set. In order to avoid overfitting, we used dropout (with $p_{dropout} = 0.5$), l_2-norm weight regularization (with regularization coefficient of 10^{-3}) and early stopping after 200 epochs without improvement in the loss of the validation set. The weights of the layers are randomly initialized as proposed by Glorot and Bengio (2010), and the biases were initialized to zero. All models were trained for a maximum of 10000 epochs.

D.4 Results and Discussion

Table D.1 show the results of the cross-validation experiments. This table shows the R^2 and r averaged across all pieces for each dataset/GMDN architecture combination.

To test whether using a multimodal distribution improves the predictive accuracy of the models, we conducted a one-way ANOVA to compare the differences in R^2 for GMDFFNNs generating predictions by taking the expected value of the distribution with 1, 2 and 5 components for

Magaloff/Chopin				
	Expected Value		Max. Component	
Components	R^2	r	R^2	r
1	0.170	0.446	0.170	0.446
2	0.204	0.483	0.191	0.469
5	0.211	0.489	0.034	0.432

Zeilinger/Beethoven				
	Expected Value		Max. Component	
Components	R^2	r	R^2	r
1	0.209	0.529	0.209	0.529
2	0.208	0.529	0.205	0.531
5	0.215	0.532	0.215	0.535

Table D.1: Predictive accuracy for MIDI velocity for GMDFFNNs with different number of components, averaged over a 5-fold cross-validation on the Magaloff/Chopin (top) and Zeilinger/Beethoven (bottom). A larger R^2 and r means better performance.

each dataset. We found a significant difference for the Magaloff/Chopin dataset ($\mathrm{F}(2, 450) = 6.21$, $p = 0.002$, $\eta^2 = 0.027$). On the other hand, we found no significant difference for the Zeilinger/Beethoven dataset ($\mathrm{F}(2, 87) = 0.04$, $p = 0.958$, $\eta^2 = 0.001$). We conducted a pairwise comparison of the models trained on the Magaloff/Chopin dataset by running three paired-samples two-tailed t-tests (i.e. 5 vs. 1, 2 vs. 1 and 5 vs. 2 components) at the $p = 0.01$ level (we used the Bonferroni correction for multiple comparisons). These results suggested that there was a significant difference for all three cases ($\mathrm{t}(150) = 11.301$, $p < 0.0001$, Cohen's $d = 0.381$ for 5 vs. 1; $\mathrm{t}(150) = 11.986$, $p < 0.0001$, Cohen's $d = 0.322$ for 2 vs. 1; $\mathrm{t}(150) = 4.505$, $p < 0.0001$, Cohen's $d = 0.064$ for 5 vs. 2).

To determine which of the two methods for generating predictions of the MDNs was more successful, we conducted paired-samples two-tailed t-tests at the $p = 0.01$ level, comparing the differences in R^2 for predictions generated by taking the expected value, and predictions generated by taking the maximal component of the mixture for each dataset. We found a significant difference for the Magaloff/Chopin dataset ($\mathrm{t}(452) = 12.833$, $p < 0.0001$, Cohen's $d = 0.454$). On the other hand, we found no difference for the Zeilinger/Beethoven dataset ($\mathrm{t}(89) = 0.301$, $p = 0.764$, Cohen's $d = 0.009$).

These results offer some evidence that the predictive accuracy of models with more than one component may be higher than the predictive accuracy of unimodal models. This finding supports the hypothesis that models of expressive performance might benefit from describing expressive parameters through multimodal distributions. However, this finding is not consistent across datasets. Still, it is likely that the number of components considered in this evaluation may not be adequate to capture the different performance strategies in the Zeilinger/Beethoven dataset. Furthermore, preliminary experiments while training the GMDNs revealed that, given the larger number of parameters when compared to deterministic neural networks, they might be more affected by changes in the architecture or the selection of hyper-parameters for training.

D.5 Visualizing Interactions Between Expressive Parameters

In this section, we present a qualitative example of how to use GMDNs to visualize the interactions (i.e. the correlations) between expressive parameters. As mentioned in Section D.1, GMDNs can be used to jointly model expressive parameters and explicitly define the correlations between expressive parameters.

Once a GMDN is trained, it is possible to visualize the interactions learned by the model by plotting the probability density function parametrized by the GMDN. Figure D.1 presents an example of the joint interactions between expressive parameters for an excerpt of Chopin's Prelude Op. 28 No. 7 in A major. In contrast to the non-sequential NBMs used in the experiments described above in Section D.3, in this example, we consider a recurrent NBM (RNBM, see Section 4.3), which uses performance codec $\mathcal{C}_{2.5}$, described in Appendix B. This performance codec describes a performance using 5 expressive parameters: two onset-wise parameters denoting the maximal MIDI velocity per each onset (referred to as MIDI velocity trend) and the logarithm of the beat period ratio for each score onset (referred to as log BPR); and three note-wise parameters describing timing deviations (referred to as timing), deviations in MIDI velocity (referred to as MIDI velocity trend) and the logarithm of the articulation ratio (referred to as log articulation). These plots were generated using GMDNs with two components trained on the Magaloff/Chopin and Beethoven datasets: An GMDFFNN for modeling note-wise expressive parameters and a recurrent GMDNs, denoted as GMDRNN, for modeling onset-wise parameters. In this figure, each color box displays the plots for the distributions comparing pairs of expressive parameters corresponding to the note marked in the score with the same color. These plots showcase an example of describing expressive parameters using multimodal distributions. This is particularly visible in the plots of timing vs. MIDI velocity deviations and log articulation vs. MIDI velocity deviations (see the bimodal distributions for A5 and C♯6, highlighted in yellow and red, respectively)

Figure D.2 presents an example of the interactions of expressive parameters for the last phrase of the same Prelude. Each of the plots in this figure corresponds to the distribution of the expressive parameters for the score onset highlighted with the same color in the score. In these plots we can see the tendency of tempo to decrease and dynamics to increase over the course of the phrase, marked by the gradual increase of log BPR (the mode of the distribution moves upwards with increasing score onsets from left to right) and the increase of MIDI velocity trend (the mode of the distribution moves to the left with each consecutive score onset). These plots support the hypothesis that there is an arch in dynamics and timing at the end of a piece, like those described by Todd (1992). An interesting observation is that the spread of the distributions tends to increase in both tempo and dynamics with each consecutive onset[3], suggesting an increase of the model's expressive freedom: the range of potential values of log BPR and MIDI velocity trend is larger towards the end of the piece (this is particularly noticeable with the last chord, highlighted in red).

Note that these examples were hand picked, and only attempt to illustrate the ability of GMDNs to capture the multimodal distribution of expressive parameters and the correlations between parameters. Future work might include a proper (quantitative) evaluation of the predictive accuracy of modeling parameters jointly or independently using GMDNs.

[3]This effect is more pronounced for log BPR than for MIDI velocity trend.

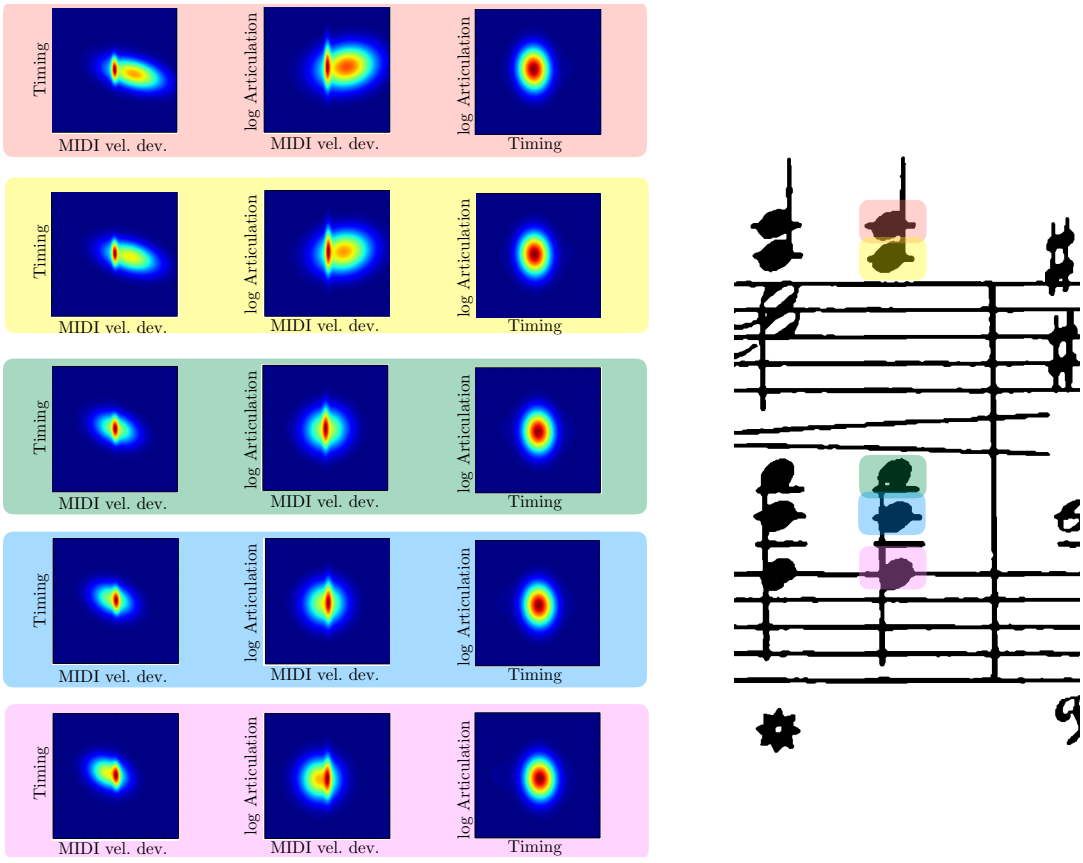


Figure D.1: Visualization of the probability density function for a GMDN modeling note-wise parameters for an excerpt of Chopin's Prelude Op. 28 No. 7 in A major. Each color box denotes the distribution for a single note (indicated with the same color in the score). The excerpt from the score is taken from Schirmer's Edition (Chopin, 1915b).

D.6 Conclusions

This appendix presented a probabilistic extension to the NBM and RNBMs discussed in Chapter 4 using MDNs. An important feature of these networks is that they allow for modeling multimodal distributions of the expressive parameters.

Our preliminary experimental evaluation suggests that using models with more than one component improves the predictive accuracy. These results support the idea that the unimodal assumptions implicitly made by the deterministic neural networks discussed in Chapter 4, as well as the linear BMs discussed in Chapter 3, might be an oversimplification. An interesting future research direction could include a more extensive evaluation of the distributions (and number of components) for modeling the parameters.

Furthermore, it would be interesting to evaluate the predictive accuracy of these models in the scenario of modeling expressive parameters jointly and independently.

An interesting direction for rendering performances would be through the use of a sampling procedure. A possible way to do so would be by combining the MDNs into Markov chain Monte Carlo sampling methods such as particle filtering.

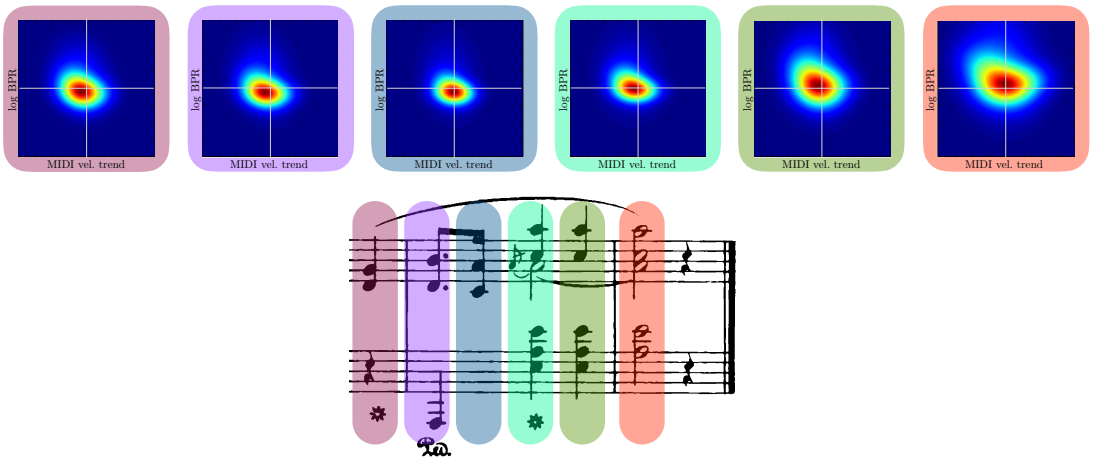


Figure D.2: Visualization of the probability density function for a GMDN modeling onset-wise parameters for the last two bars of Chopin's Prelude Op. 28 No. 7 in A major. Each color box denotes the distribution for a single onset (indicated with the same color in the score). Thin grey horizontal and vertical gridlines were added to the plots of the distributions to better compare the location of the mode of the distribution. The excerpt from the score is taken from Schirmer's Edition (Chopin, 1915b).

E Constructing and Training Linear Models

In this appendix we provide some theoretical results and derivations for constructing and training the linear models described in Chapter 3.

E.1 Standard Least Squares

In this appendix, a detailed derivation of the weights for linear regression in the least squares (LS) sense, discussed in Section 3.3.3 as the basis for the linear basis model (LBM), is provided. To unclutter notation, we drop the subscript $\boldsymbol{\mathcal{T}}$ from $\mathbf{t}_{\boldsymbol{\mathcal{T}}}$ and $\boldsymbol{\Phi}_{\boldsymbol{\mathcal{T}}}$ defined in Equation (3.21).

The weights computed in the LS sense are given by

$$\mathbf{w}_{ls} = \operatorname*{argmin}_{\mathbf{w}} \mathcal{L}_{se}(\mathbf{w}), \tag{E.1}$$

where $\mathcal{L}_{se}$, the squared error, is written as

$$\begin{aligned}\mathcal{L}_{se}(\mathbf{w}) &= \|\mathbf{t} - \boldsymbol{\Phi}\mathbf{w}\|^2 \\ &= (\mathbf{t} - \boldsymbol{\Phi}\mathbf{w})^{\mathrm{T}}(\mathbf{t} - \boldsymbol{\Phi}\mathbf{w}).\end{aligned} \tag{E.2}$$

Computing the gradient of the cost function with respect to the weights results in

$$\frac{\partial}{\partial \mathbf{w}}\mathcal{L}_{se}(\mathbf{w}) = -2\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t} + 2\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\mathbf{w}. \tag{E.3}$$

The weights that minimize the cost function are given by the solutions to $\frac{\partial}{\partial \mathbf{w}}\mathcal{L}_{se}(\mathbf{w}) = \mathbf{0}$, i.e.

$$\mathbf{w}_{ls} = \boldsymbol{\Phi}^{\dagger}\mathbf{t}, \tag{E.4}$$

where

$$\boldsymbol{\Phi}^{\dagger} = \left(\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\right)^{-1}\boldsymbol{\Phi}^{\mathrm{T}} \tag{E.5}$$

is the Moore-Penrose pseudo-inverse of the matrix $\boldsymbol{\Phi}$.

Under the assumption of a Gaussian conditional distribution of $\mathbf{t}$ given $\mathbf{w}$ (as in Equation (3.26)), the LS regression is analytically equivalent to ML estimation of $\mathbf{w}$. The conditional log-likelihood of $\mathbf{t}$ given $\mathbf{w}$ is given by

$$\log p(\mathbf{t} \mid \mathbf{w}) = \frac{N}{2}\log(\beta) - \frac{N}{2}\log(2\pi) - \frac{\beta}{2}(\mathbf{t} - \boldsymbol{\Phi}\mathbf{w})^{\mathrm{T}}(\mathbf{t} - \boldsymbol{\Phi}\mathbf{w}). \tag{E.6}$$

The gradient of the above log-likelihood is proportional to the gradient of the LS cost function, i.e.

$$\frac{\partial}{\partial \mathbf{w}}\log p(\mathbf{t} \mid \mathbf{w}) = \beta\boldsymbol{\Phi}^{\mathrm{T}}\mathbf{t} - \beta\boldsymbol{\Phi}^{\mathrm{T}}\boldsymbol{\Phi}\mathbf{w} = -\frac{\beta}{2}\frac{\partial}{\partial \mathbf{w}}\mathcal{L}_{se}(\mathbf{w}). \tag{E.7}$$

Therefore, computing the solution that maximizes the conditional log-likelihood is equivalent to computing the solution that minimizes the squared error. Hence, the weights computed using ML are given by

$$\mathbf{w}_{ml} = \mathbf{w}_{ls} = \mathbf{\Phi}^{\dagger}\mathbf{t}. \tag{E.8}$$

E.2 Regularized Least Squares

In this appendix, we provide a detailed derivation of the weights for the regularized linear regression using a quadratic regularization term.

Using a quadratic regularization term, the regularized squared error $\mathcal{L}_{se\ reg}$ can be written as

$$\mathcal{L}_{se\ reg}(\mathbf{w}) = \|\mathbf{t} - \mathbf{\Phi}\mathbf{w}\|^2 \ + \lambda_{l2}\|\mathbf{w}\|^2, \tag{E.9}$$

where $\lambda_{l2} \in \mathbb{R}_{\geq 0}$ is called the *l_2-norm weight regularization coefficient* (or simply *regularization coefficient*).

Computing the gradient of the cost function with respect to the weights results in

$$\frac{\partial}{\partial \mathbf{w}}\mathcal{L}_{se\ reg}(\mathbf{w}) = -2\mathbf{\Phi}^{\mathrm{T}}\mathbf{t} + 2\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\mathbf{w} + 2\lambda_{l2}\mathbf{w}. \tag{E.10}$$

The weights that minimize the cost function are given by the solutions to $\frac{\partial}{\partial \mathbf{w}}\mathcal{L}_{ls}(\mathbf{w}) = \mathbf{0}$, which can be written as

$$\mathbf{w}_{ls\ reg} = \left(\lambda_{l2}\mathbf{I} + \mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1}\mathbf{\Phi}^{\mathrm{T}}\mathbf{t}. \tag{E.11}$$

Note that when $\lambda_{l2} \to 0$, the solution of regularized LS converges to the standard LS solution from Equation (E.4).

E.3 Derivation of the Weights for Bayesian Linear Regression Using a Gaussian Prior

In this appendix, we provide a detailed derivation of the weights for Bayesian linear regression described in Section 3.4.2.

The joint probability distribution of the expressive parameters $\mathbf{t}$ and the weights $\mathbf{w}$ given by

$$p(\mathbf{t}, \mathbf{w}) = p(\mathbf{t} \mid \mathbf{w})p(\mathbf{w}), \tag{E.12}$$

where $p(\mathbf{w})$ is the conjugate prior distribution of the weights (Bishop, 2006). As previously stated in Section 3.4.2, we assume this prior probability distribution to be Gaussian, i.e.

$$p(\mathbf{w}) = \mathcal{N}(\mathbf{w} \mid \mathbf{m}_0, \mathbf{S}_0), \tag{E.13}$$

where $\mathbf{m}_0$ is the prior mean and $\mathbf{S}_0$ is the prior covariance matrix.

Using Bayes' Theorem, the posterior probability distribution of $\mathbf{w}$ given $\mathbf{t}$ (in log-domain) is given by

$$\log p(\mathbf{w} \mid \mathbf{t}) = \log p(\mathbf{t} \mid \mathbf{w}) + \log p(\mathbf{w}) - \underbrace{\log \int_{\mathbf{w}} p(\mathbf{t} \mid \mathbf{w}) p(\mathbf{w}) d\mathbf{w}}_{\text{does not depend on } \mathbf{w}}. \tag{E.14}$$

The weights for Bayesian linear regression can be computed using MAP estimation, i.e.

$$\begin{aligned} \mathbf{w}_{map} &= \underset{\mathbf{w}}{\operatorname{argmax}} \left(\log p(\mathbf{w} \mid \mathbf{t}) \right) \\ &= \underset{\mathbf{w}}{\operatorname{argmax}} \left(\underbrace{\log p(\mathbf{t} \mid \mathbf{w}) + \log p(\mathbf{w})}_{\mathcal{L}_{map}(\mathbf{w})} \right). \end{aligned} \tag{E.15}$$

Let us analyze every term of the cost function $\mathcal{L}_{map}$ separately. The log-posterior probability of the expressive parameter given the weights can be written as

$$\log p(\mathbf{t} \mid \mathbf{w}) = \frac{N}{2} \log(\beta) - \frac{N}{2} \log(2\pi) - \frac{\beta}{2} \left(\mathbf{t}^{\mathrm{T}} \mathbf{t} - 2\mathbf{t}^{\mathrm{T}} \mathbf{\Phi} \mathbf{w} + \mathbf{w}^{\mathrm{T}} \mathbf{\Phi}^{\mathrm{T}} \mathbf{\Phi} \mathbf{w} \right). \tag{E.16}$$

The log-prior probability of the weights is given by

$$\log p(\mathbf{w}) = -\frac{M}{2} \log(2\pi) - \frac{1}{2} \log(\det(\mathbf{S}_0)) - \frac{1}{2} \left((\mathbf{w} - \mathbf{m}_0)^{\mathrm{T}} \mathbf{S}_0^{-1} (\mathbf{w} - \mathbf{m}_0) \right). \tag{E.17}$$

The gradient with respect to the weights of the log likelihood $\mathcal{L}_{map}(\mathbf{w})$ results in

$$\begin{aligned} \frac{\partial}{\partial \mathbf{w}} \mathcal{L}_{map}(\mathbf{w}) &= \beta \mathbf{t}^{\mathrm{T}} \mathbf{\Phi} - \beta \mathbf{\Phi}^{\mathrm{T}} \mathbf{\Phi} \mathbf{w} - \frac{1}{2} \left(\mathbf{S}_0 + \mathbf{S}_0^{-1^{\mathrm{T}}} \right) (\mathbf{w} - \mathbf{m}_0) \\ &= \beta \mathbf{t}^{\mathrm{T}} \mathbf{\Phi} - \beta \mathbf{\Phi}^{\mathrm{T}} \mathbf{\Phi} \mathbf{w} - \mathbf{S}_0^{-1} (\mathbf{w} - \mathbf{m}_0). \end{aligned} \tag{E.18}$$

The weights that maximize the cost function are the solution to $\frac{\partial}{\partial \mathbf{w}} \mathcal{L}_{map}(\mathbf{w}) = \mathbf{0}$, i.e.

$$\mathbf{w}_{map} = \left(\beta \mathbf{\Phi}^{\mathrm{T}} \mathbf{\Phi} + \mathbf{S}_0^{-1} \right)^{-1} \left(\beta \mathbf{\Phi}^{\mathrm{T}} \mathbf{t} + \mathbf{S}_0^{-1} \mathbf{m}_0 \right). \tag{E.19}$$

In case $\mathbf{m}_0 = \mathbf{0}$ and $\mathbf{S}_0 = \alpha^{-1} \mathbf{I}$, the solution simplifies to

$$\mathbf{w}_{map} = \left(\frac{\alpha}{\beta} \mathbf{I} + \mathbf{\Phi}^{\mathrm{T}} \mathbf{\Phi} \right)^{-1} \mathbf{\Phi}^{\mathrm{T}} \mathbf{t}. \tag{E.20}$$

E.4 Estimation of the Parameters of the Prior Distribution of $p(\mathbf{w})$ Using the Expectation-Maximization Algorithm

This appendix describes the estimation of the mean value $\mathbf{m}_0$ and covariance matrix $\mathbf{S}_0$ of the prior distribution of the weights $p(\mathbf{w})$ and the noise precision β of the conditional probability distribution $p(\mathbf{t} \mid \mathbf{w})$ using the expectation-maximization (EM) algorithm.

For the expectation step (E-step) we need to compute the conditional expectation of the log-likelihood $\mathcal{L}_{map}$, given as in Equation (E.15), with respect to the posterior probability of the

weights $\mathbf{w}$ given the expressive parameter $\mathbf{t}$. Using Bayes' theorem, the posterior probability of $\mathbf{w}$ given $\mathbf{t}$ is given by[1]

$$p(\mathbf{w} \mid \mathbf{t}) = \mathcal{N}(\mathbf{w} \mid \mathbf{m}_N, \mathbf{S}_N), \tag{E.21}$$

where $\mathbf{m}_N$ is the posterior mean and $\mathbf{S}_N$ is the posterior covariance matrix, calculated as

$$\mathbf{m}_N = \mathbf{S}_N \left(\mathbf{S}_0^{-1}\mathbf{m}_0 + \beta\mathbf{\Phi}^{\mathrm{T}}\mathbf{t}\right) \qquad \text{and} \qquad \mathbf{S}_N^{-1} = \mathbf{S}_0^{-1} + \beta\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}. \tag{E.22}$$

The conditional expectation of the log-likelihood function results in

$$\mathbb{E}\left\{\mathcal{L}_{map}(\mathbf{w})\right\} = \mathbb{E}\left\{\log p(\mathbf{t} \mid \mathbf{w})\right\} + \mathbb{E}\left\{\log p(\mathbf{w})\right\} \tag{E.23}$$

Let us compute the conditional expected values from the above equation individually

$$\begin{aligned}\mathbb{E}\left\{\log p(\mathbf{t} \mid \mathbf{w})\right\} &= \int_{\mathbf{w}} \log p(\mathbf{t} \mid \mathbf{w}) p(\mathbf{w} \mid \mathbf{t}) d\mathbf{w} \\ &= \frac{N}{2}\log(\beta) - \frac{N}{2}\log(2\pi) - \frac{\beta}{2}\left(\mathbf{t}^{\mathrm{T}}\mathbf{t} - 2\mathbf{t}^{\mathrm{T}}\mathbf{\Phi}\underbrace{\mathbb{E}\left\{\mathbf{w}\right\}}_{=\mathbf{m}_N} + \underbrace{\mathbb{E}\left\{\mathbf{w}^{\mathrm{T}}\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\mathbf{w}\right\}}_{**}\right).\end{aligned} \tag{E.24}$$

In order to compute the term denoted as $**$, we consider the eigenvalue decomposition of $\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}$. Since this product is a symmetric matrix, we can rewrite it as

$$\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi} = \sum_{i=1}^{M} \nu_i \mathbf{u}_i \mathbf{u}_i^{\mathrm{T}}, \tag{E.25}$$

where ν_i is the i-th eigenvalue and $\mathbf{u}_i$ the i-th eigenvector of $\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}$, respectively. Substituting the above equation in $**$ results in

$$\begin{aligned}** &= \mathbb{E}\left\{\mathbf{w}^{\mathrm{T}}\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\mathbf{w}\right\} \\ &= \mathbb{E}\left\{\sum_{i=1}^{M} \nu_i \mathbf{w}^{\mathrm{T}}\mathbf{u}_i\mathbf{u}_i^{\mathrm{T}}\mathbf{w}\right\} \\ &= \mathbb{E}\left\{\sum_{i=1}^{M} \nu_i \mathbf{u}_i^{\mathrm{T}}\mathbf{w}\mathbf{w}^{\mathrm{T}}\mathbf{u}_i\right\} \\ &= \sum_{i=1}^{M} \nu_i \mathbf{u}_i^{\mathrm{T}} \underbrace{\mathbb{E}\left\{\mathbf{w}\mathbf{w}^{\mathrm{T}}\right\}}_{=\mathbf{m}_N\mathbf{m}_N^{\mathrm{T}}+\mathbf{S}_N} \mathbf{u}_i \\ &= \mathbf{m}_N^{\mathrm{T}}\left(\sum_{i=1}^{M} \nu_i \mathbf{u}_i\mathbf{u}_i^{\mathrm{T}}\right)\mathbf{m}_N + \sum_{i=1}^{M} \nu_i \mathbf{u}_i^{\mathrm{T}}\mathbf{S}_N\mathbf{u}_i \\ &= \mathbf{m}_N^{\mathrm{T}}\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\mathbf{m}_N + \underbrace{\sum_{i=1}^{M} \nu_i \mathbf{u}_i^{\mathrm{T}}\mathbf{S}_N\mathbf{u}_i}_{***}.\end{aligned} \tag{E.26}$$

[1] See Equation (H.4) in Appendix H.1.

We can use properties of the trace[2] to rewrite $***$ as

$$\begin{aligned}
*** &= \sum_{i=1}^{M} \nu_i \mathbf{u}_i^{\mathrm{T}} \mathbf{S}_N \mathbf{u}_i \\
&= \mathrm{Tr}\left(\sum_{i=1}^{M} \nu_i \mathbf{u}_i^{\mathrm{T}} \mathbf{S}_N \mathbf{u}_i \right) \\
&= \mathrm{Tr}\left(\sum_{i=1}^{M} \nu_i \mathbf{S}_N \mathbf{u}_i \mathbf{u}_i^{\mathrm{T}} \right) \\
&= \mathrm{Tr}\left(\mathbf{S}_N \sum_{i=1}^{M} \nu_i \mathbf{u}_i \mathbf{u}_i^{\mathrm{T}} \right) \\
&= \mathrm{Tr}\left(\mathbf{S}_N \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi} \right) \\
&= \mathrm{Tr}\left(\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi} \mathbf{S}_N \right).
\end{aligned} \tag{E.27}$$

Substituting the above result in $**$, we get

$$** = \mathbf{m}_N^{\mathrm{T}} \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi} \mathbf{m}_N + \mathrm{Tr}\left(\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi} \mathbf{S}_N \right). \tag{E.28}$$

Using the above equation, Equation (E.24) can be rewritten as

$$\begin{aligned}
\mathbb{E}\left\{ \log p(\mathbf{t} \mid \mathbf{w}) \right\} =& \frac{N}{2} \log(\beta) - \frac{N}{2} \log(2\pi) - \frac{\beta}{2} \mathbf{t}^{\mathrm{T}} \mathbf{t} \\
&+ \beta \mathbf{t}^{\mathrm{T}} \boldsymbol{\Phi} \mathbf{m}_N - \frac{\beta}{2} \left(\mathbf{m}_N \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi} \mathbf{m}_N + \mathrm{Tr}\left(\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi} \mathbf{S}_N \right) \right).
\end{aligned} \tag{E.29}$$

In a similar fashion, the conditional expectation of the prior probability is given by

$$\begin{aligned}
\mathbb{E}\left\{ \log p(\mathbf{w}) \right\} &= -\frac{M}{2} \log(2\pi) - \frac{1}{2} \log(\det(\mathbf{S}_0)) \\
&\quad - \frac{1}{2} \left(\underbrace{\mathbb{E}\left\{ \mathbf{w}^{\mathrm{T}} \mathbf{S}_0^{-1} \mathbf{w} \right\}}_{= \mathbf{m}_N^{\mathrm{T}} \mathbf{S}_0^{-1} \mathbf{m}_N + \mathrm{Tr}\left(\mathbf{S}_0^{-1} \mathbf{S}_N \right)} - \mathbf{m}_0^{\mathrm{T}} \left(\mathbf{S}_0^{-1^{\mathrm{T}}} + \mathbf{S}_0^{-1} \right) \underbrace{\mathbb{E}\left\{ \mathbf{w} \right\}}_{= \mathbf{m}_N} + \mathbf{m}_0^{\mathrm{T}} \mathbf{S}_0^{-1} \mathbf{m}_0 \right) \\
&= -\frac{M}{2} \log(2\pi) - \frac{1}{2} \log(\det(\mathbf{S}_0)) - \frac{1}{2} (\mathbf{m}_N - \mathbf{m}_0)^{\mathrm{T}} \mathbf{S}_0^{-1} (\mathbf{m}_N - \mathbf{m}_0) \\
&\quad - \frac{1}{2} \mathrm{Tr}\left(\mathbf{S}_0^{-1} \mathbf{S}_N \right).
\end{aligned} \tag{E.30}$$

Substituting in Equation (E.29) and Equation (E.30) in Equation (E.23), the expectation of the cost function $\mathcal{L}_{map}$ results in

$$\begin{aligned}
\mathbb{E}\left\{ \mathcal{L}_{map}(\mathbf{w}) \right\} =& \frac{N}{2} \log(\beta) - \frac{N+M}{2} \log(2\pi) - \frac{1}{2} \log(\det(\mathbf{S}_0)) \\
&- \frac{\beta}{2} \mathbf{t}^{\mathrm{T}} \mathbf{t} + \beta \mathbf{t}^{\mathrm{T}} \boldsymbol{\Phi} \mathbf{m}_N - \frac{\beta}{2} \mathrm{Tr}(\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi} \mathbf{S}_N) - \frac{\beta}{2} \mathbf{m}_N^{\mathrm{T}} \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi} \mathbf{m}_N \\
&- \frac{1}{2} (\mathbf{m}_N - \mathbf{m}_0)^{\mathrm{T}} \mathbf{S}_0^{-1} (\mathbf{m}_N - \mathbf{m}_0) - \frac{1}{2} \mathrm{Tr}\left(\mathbf{S}_0^{-1} \mathbf{S}_N \right).
\end{aligned} \tag{E.31}$$

[2] See Equation (H.11) and Equation (H.12) in Appendix H.2.

We can proceed to the maximization step (M-step). In order to avoid overfitting of the mean value $\mathbf{m}_0$ caused by $\|\mathbf{S}_0\| \to 0$ and $\beta \to \infty$, thus, rendering the priors non informative, we need to introduce a constraint on $\mathbf{S}_0$. A natural bound for the estimated covariance is the Cramér-Rao lower bound[3]

$$\|\mathbf{S}_0\|^2 \geq \left\| \left(\beta \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi}\right)^{-1} \right\|^2. \tag{E.36}$$

Using the Frobenius norm[4], we can express this constraint as

$$-g(\beta, \mathbf{S}_0) = \mathrm{Tr}\left(\mathbf{S}_0 \mathbf{S}_0^{\mathrm{T}}\right) - \frac{1}{\beta^2} \mathrm{Tr}\left(\left(\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi}\right)^{-1} \left(\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi}\right)^{-1^{\mathrm{T}}}\right) \geq 0. \tag{E.37}$$

We proceed to maximize $\mathbb{E}\{\mathcal{L}_{map}\}$ subject to the above constraint using the method of Lagrange multipliers. The Lagrangian function is given by

$$\begin{aligned}
\mathcal{L}_{em}(\beta, \mathbf{m}_0, \mathbf{S}_0, \kappa) =& \frac{N}{2} \log(\beta) - \frac{N+M}{2} \log(2\pi) - \frac{1}{2} \log\left(\det(\mathbf{S}_0)\right) \\
&- \frac{\beta}{2} \mathbf{t}^{\mathrm{T}} \mathbf{t} + \beta \mathbf{t}^{\mathrm{T}} \boldsymbol{\Phi} \mathbf{m}_N - \frac{\beta}{2} \mathrm{Tr}\left(\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi} \mathbf{S}_N\right) - \frac{\beta}{2} \mathbf{m}_N^{\mathrm{T}} \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi} \mathbf{m}_N \\
&- \frac{1}{2} \left(\mathbf{m}_N - \mathbf{m}_0\right)^{\mathrm{T}} \mathbf{S}_0^{-1} (\mathbf{m}_N - \mathbf{m}_0) - \frac{1}{2} \mathrm{Tr}\left(\mathbf{S}_0^{-1} \mathbf{S}_N\right) \\
&+ \kappa \left(\mathrm{Tr}\left(\mathbf{S}_0 \mathbf{S}_0^{\mathrm{T}}\right) - \frac{1}{\beta^2} \mathrm{Tr}\left(\left(\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi}\right)^{-1} \left(\boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi}\right)^{-1^{\mathrm{T}}}\right)\right).
\end{aligned} \tag{E.38}$$

Since this optimization problem has an inequality constraint, we consider the Karush-Kuhn-Tucker (KKT) conditions (Boyd and Vandenberghe, 2004). These conditions can be stated as follows:

$$\begin{aligned}
&\text{Stationarity:} && \frac{\partial}{\partial \beta} \mathcal{L}_{em} = 0, \frac{\partial}{\partial \mathbf{m}_0} \mathcal{L}_{em} = \mathbf{0}, \frac{\partial}{\partial \mathbf{S}_0} \mathcal{L}_{em} = \mathbf{0} \\
&\text{Primal feasibility:} && g(\beta, \mathbf{S}_0) \leq 0 \\
&\text{Dual feasibility:} && \kappa \geq 0 \\
&\text{Complimentary slackness:} && \kappa g(\beta, \mathbf{S}_0) = 0,
\end{aligned} \tag{E.39}$$

[3] According to the Cramér-Rao lower bound (CRLB) theorem, the covariance of $\mathbf{w}$ is bounded by

$$\mathrm{cov}(\mathbf{w}) \geq \mathrm{FIM}_{\mathbf{w}}^{-1}, \tag{E.32}$$

where $\mathrm{FIM}_{\mathbf{w}}$ is the *Fisher information matrix*, given by

$$\mathrm{FIM}_{\mathbf{w}} = -\mathbb{E}\left\{ \frac{\partial^2}{\partial \mathbf{w}^2} \log p(\mathbf{t} \mid \mathbf{w}) \right\}. \tag{E.33}$$

Using Equation (E.7), we can rewrite the above equation as

$$\begin{aligned}
\mathrm{FIM}_{\mathbf{w}} &= -\mathbb{E}\left\{ \frac{\partial}{\partial \mathbf{w}} \left(\beta \boldsymbol{\Phi}^{\mathrm{T}} \mathbf{t} - \beta \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi} \mathbf{w}\right) \right\} \\
&= -\mathbb{E}\left\{ -\beta \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi} \right\} \\
&= \beta \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi}.
\end{aligned} \tag{E.34}$$

Therefore, the CRLB results in

$$\mathrm{cov}(\mathbf{w}) \geq \left(\beta \boldsymbol{\Phi}^{\mathrm{T}} \boldsymbol{\Phi}\right)^{-1}. \tag{E.35}$$

[4] See Equation (H.17) in Appendix H.2.

where κ is the KKT multiplier.

Since the covariance matrix $\mathbf{S}_0$ must be symmetric and positive definite[5]. Therefore we can express $\mathbf{S}_0$ as

$$\mathbf{S}_0 = \mathbf{L}\mathbf{L}^{\mathrm{T}}, \tag{E.40}$$

where $\mathbf{L} \in \mathbb{R}^{K\times K}$ is an invertible matrix[6]. We can rewrite the cost function in terms of $\mathbf{L}$ as

$$\begin{aligned}\mathcal{L}_{em}(\beta, \mathbf{m}_0, \mathbf{L}, \kappa) =& \frac{N}{2}\log(\beta) - \frac{N+M}{2}\log(2\pi) - \frac{1}{2}\log(\det(\mathbf{L}\mathbf{L}^{\mathrm{T}})) \\ &- \frac{\beta}{2}\mathbf{t}^{\mathrm{T}}\mathbf{t} + \beta\mathbf{t}^{\mathrm{T}}\mathbf{\Phi}\mathbf{m}_N - \frac{\beta}{2}\operatorname{Tr}(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\mathbf{S}_N) - \frac{\beta}{2}\mathbf{m}_N^{\mathrm{T}}\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\mathbf{m}_N \\ &- \frac{1}{2}(\mathbf{m}_N - \mathbf{m}_0)^{\mathrm{T}}\mathbf{L}^{-1^{\mathrm{T}}}\mathbf{L}^{-1}(\mathbf{m}_N - \mathbf{m}_0) - \frac{1}{2}\operatorname{Tr}\left(\mathbf{L}^{-1^{\mathrm{T}}}\mathbf{L}^{-1}\mathbf{S}_N\right) \\ &+ \kappa\left(\operatorname{Tr}\left(\mathbf{L}\mathbf{L}^{\mathrm{T}}\mathbf{L}^{\mathrm{T}}\mathbf{L}\right) - \frac{1}{\beta^2}\operatorname{Tr}\left(\left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1}\left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1^{\mathrm{T}}}\right)\right)\end{aligned} \tag{E.41}$$

The partial derivative of the cost function with respect to β is given by

$$\begin{aligned}\frac{\partial\mathcal{L}_{em}}{\partial\beta} =& \frac{N}{2\beta} - \frac{1}{2}\mathbf{t}^{\mathrm{T}}\mathbf{t} + \mathbf{t}^{\mathrm{T}}\mathbf{\Phi}\mathbf{m}_N - \frac{1}{2}\operatorname{Tr}\left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\mathbf{S}_N\right) - \frac{1}{2}\mathbf{m}_N\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\mathbf{m}_N \\ &+ \frac{2\kappa}{\beta^3}\operatorname{Tr}\left(\left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1}\left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1^{\mathrm{T}}}\right).\end{aligned} \tag{E.42}$$

The gradient of the cost function with respect to $\mathbf{m}_0$ is given by

$$\frac{\partial\mathcal{L}_{em}}{\partial\mathbf{m}_0} = \mathbf{L}^{-1^{\mathrm{T}}}\mathbf{L}^{-1}(\mathbf{m}_N - \mathbf{m}_0). \tag{E.43}$$

For the sake of clarity of the calculation of the gradient of the cost function with respect to $\mathbf{L}$, we compute individually the partial derivatives of every non constant term of the cost function

[5] By definition, covariance matrices are positive semi-definite. Furthermore, an invertible positive semidefinite matrix is positive definite (Burden and Faires, 2001), and therefore, since we assume that $\mathbf{S}_0$ is invertible, $\mathbf{S}_0$ is positive definite.

[6] If $\mathbf{L}$ is a lower triangular matrix with non-negative elements in the diagonal, this is the Cholesky decomposition (Burden and Faires, 2001), nevertheless a particular structure of matrix $\mathbf{L}$ is not required for $\mathbf{S}_0^{-1}$ to be positive definite (See Section 9.6.6 in (Petersen and Pedersen, 2012)).

that has a dependency in $\mathbf{L}$:

$$\begin{aligned}\frac{\partial}{\partial \mathbf{L}} \log(\det(\mathbf{L}\mathbf{L}^{\mathrm{T}})) &= \frac{\partial}{\partial \mathbf{L}} \log(\det(\mathbf{L}) \det(\mathbf{L}^{\mathrm{T}})) \\ &= 2\frac{\partial}{\partial \mathbf{L}} \log((\det(\mathbf{L}))) \\ &= 2\mathbf{L}^{-1^{\mathrm{T}}}, \end{aligned} \tag{E.44}$$

$$\begin{aligned}\frac{\partial}{\partial \mathbf{L}} (\mathbf{m}_N - \mathbf{m}_0)^{\mathrm{T}} (\mathbf{L}\mathbf{L})^{-1} (\mathbf{m}_N - \mathbf{m}_0) &= -(\mathbf{L}\mathbf{L}^{\mathrm{T}})^{-1} \frac{\partial (\mathbf{m}_N - \mathbf{m}_0)^{\mathrm{T}} \mathbf{L}\mathbf{L}^{\mathrm{T}} (\mathbf{m}_N - \mathbf{m}_0)}{\partial \mathbf{L}} (\mathbf{L}\mathbf{L}^{\mathrm{T}})^{-1} \\ &= -2(\mathbf{m}_N - \mathbf{m}_0)^{\mathrm{T}} (\mathbf{m}_N - \mathbf{m}_0) (\mathbf{L}\mathbf{L}^{\mathrm{T}})^{-1} \mathbf{L} (\mathbf{L}\mathbf{L}^{\mathrm{T}})^{-1}, \end{aligned} \tag{E.45}$$

$$\begin{aligned}\frac{\partial}{\partial \mathbf{L}} \operatorname{Tr}\left(\left(\mathbf{L}\mathbf{L}^{\mathrm{T}}\right)^{-1} \mathbf{S}_N \right) &= \frac{\partial}{\partial \mathbf{L}} \operatorname{Tr}\left(\mathbf{S}_N \left(\mathbf{L}\mathbf{L}^{\mathrm{T}}\right)^{-1^{\mathrm{T}}} \right) \\ &= \frac{\partial}{\partial \mathbf{L}} \operatorname{Tr}\left(\mathbf{S}_N \left(\mathbf{L}^{\mathrm{T}}\mathbf{L}\right)^{-1} \right) \\ &= \frac{\partial}{\partial \mathbf{L}} \operatorname{Tr}\left(\left(\mathbf{L}^{\mathrm{T}}\mathbf{I}\mathbf{L}\right)^{-1} \mathbf{S}_N \right) \\ &= -2\left(\mathbf{I}\mathbf{L} \left(\mathbf{L}^{\mathrm{T}}\mathbf{I}\mathbf{L}\right)^{-1} \right) \mathbf{S}_N \left(\mathbf{L}^{\mathrm{T}}\mathbf{I}\mathbf{L}\right)^{-1} \\ &= -2\mathbf{L}^{-1^{\mathrm{T}}} \mathbf{S}_N \left(\mathbf{L}^{\mathrm{T}}\mathbf{L}\right)^{-1}, \end{aligned} \tag{E.46}$$

$$\begin{aligned}\frac{\partial}{\partial \mathbf{L}} \operatorname{Tr}\left(\mathbf{L}\mathbf{L}^{\mathrm{T}}\mathbf{L}^{\mathrm{T}}\mathbf{L}\right) &= \frac{\partial}{\partial \mathbf{L}} \operatorname{Tr}\left(\left(\mathbf{L}\mathbf{L}^{\mathrm{T}}\right)^{\mathrm{T}} \left(\mathbf{L}^{\mathrm{T}}\mathbf{L}\right) \right) \\ &= \frac{\partial}{\partial \mathbf{L}} \operatorname{Tr}\left(\mathbf{I}\mathbf{L}^{\mathrm{T}}\mathbf{I}\mathbf{L}\mathbf{L}^{\mathrm{T}}\mathbf{I}\mathbf{L}\mathbf{I}\right) \\ &= \mathbf{I}\mathbf{L}\mathbf{L}^{\mathrm{T}}\mathbf{I}\mathbf{L}\mathbf{I}\mathbf{I}^{\mathrm{T}} + \mathbf{I}\mathbf{L}\mathbf{I}\mathbf{I}^{\mathrm{T}}\mathbf{L}^{\mathrm{T}}\mathbf{I}\mathbf{L} \\ &\quad + \mathbf{I}\mathbf{L}\mathbf{I}\mathbf{I}\mathbf{L}^{\mathrm{T}}\mathbf{I}\mathbf{L} + \mathbf{I}^{\mathrm{T}}\mathbf{L}\mathbf{L}^{\mathrm{T}}\mathbf{I}^{\mathrm{T}}\mathbf{L}\mathbf{I}\mathbf{I} \\ &= 3\mathbf{L}\mathbf{L}^{\mathrm{T}}\mathbf{L}. \end{aligned} \tag{E.47}$$

Using the above results, the partial derivative of the cost function with respect to $\mathbf{L}$ is given by

$$\begin{aligned}\frac{\partial \mathcal{L}_{em}}{\partial \mathbf{L}} =& -\mathbf{L}^{-1^{\mathrm{T}}} + (\mathbf{m}_N - \mathbf{m}_0)^{\mathrm{T}} (\mathbf{m}_N - \mathbf{m}_0) (\mathbf{L}\mathbf{L}^{\mathrm{T}})^{-1} \mathbf{L} (\mathbf{L}\mathbf{L}^{\mathrm{T}})^{-1} \\ &+ \mathbf{L}^{-1^{\mathrm{T}}} \mathbf{S}_N \left(\mathbf{L}^{\mathrm{T}}\mathbf{L}\right)^{-1} + 3\kappa \mathbf{L}\mathbf{L}^{\mathrm{T}}\mathbf{L} \end{aligned} \tag{E.48}$$

Substituting Equations (E.42), (E.43) and (E.48) in the KKT conditions from Equation (E.39), maximizing the cost function is equivalent to solve the following system of equations for β, $\mathbf{m}_0$,

$\mathbf{S}_0$ and κ:

$$a\beta^3 - N\beta^2 - 4\kappa\left\|\left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1}\right\|^2 = 0 \tag{E.49}$$

$$\mathbf{L}^{-1^{\mathrm{T}}}\mathbf{L}^{-1}(\mathbf{m}_N - \mathbf{m}_0) = \mathbf{0} \tag{E.50}$$

$$\begin{aligned}&-\mathbf{L}^{-1^{\mathrm{T}}} + \mathbf{L}^{-1^{\mathrm{T}}}\mathbf{S}_N\left(\mathbf{L}^{\mathrm{T}}\mathbf{L}\right)^{-1} + 3\kappa\mathbf{L}\mathbf{L}^{\mathrm{T}}\mathbf{L} - \mathbf{L}^{-1^{\mathrm{T}}}\\ &+(\mathbf{m}_N - \mathbf{m}_0)^{\mathrm{T}}(\mathbf{m}_N - \mathbf{m}_0)(\mathbf{L}\mathbf{L}^{\mathrm{T}})^{-1}\mathbf{L}(\mathbf{L}\mathbf{L}^{\mathrm{T}})^{-1} = \mathbf{0}_M\end{aligned} \tag{E.51}$$

$$\kappa\left(\frac{1}{\beta^2}\left\|\left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1}\right\|^2 - \mathrm{Tr}\left(\mathbf{L}\mathbf{L}^{\mathrm{T}}\mathbf{L}^{\mathrm{T}}\mathbf{L}\right)\right) = 0 \tag{E.52}$$

$$\left\|\mathbf{L}\mathbf{L}^{\mathrm{T}}\right\|^2 - \frac{1}{\beta^2}\left\|\left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1}\right\|^2 \geq 0 \tag{E.53}$$

$$\kappa \geq 0, \tag{E.54}$$

where $\mathbf{0}_M$ represents the null matrix in $\mathbb{R}^{M\times M}$, and a is constant factor given by

$$a = \mathbf{t}^{\mathrm{T}}\mathbf{t} - 2\mathbf{t}^{\mathrm{T}}\mathbf{\Phi}\mathbf{m}_N + \mathrm{Tr}\left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\mathbf{S}_N\right) + \mathbf{m}_N^{\mathrm{T}}\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\mathbf{m}_N. \tag{E.55}$$

In order to satisfy the KKT constraints (Equation (E.53) and Equation (E.54)) we have two alternatives, namely $\kappa = 0$ or $\kappa > 0$ and $\left\|\mathbf{L}\mathbf{L}^{\mathrm{T}}\right\|^2 = \frac{1}{\beta^2}\left\|\left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1}\right\|^2$. In the following, we consider the two cases individually:

1. $\kappa = 0$

   By substituting κ in the KKT conditions, it is easy to see that the parameters that maximizes the cost function are given by

   $$\beta = \frac{N}{a} \tag{E.56}$$

   $$\mathbf{m}_0 = \mathbf{m}_N \tag{E.57}$$

   $$\mathbf{L}\mathbf{L}^{\mathrm{T}} = \mathbf{S}_N. \tag{E.58}$$

2. $\kappa > 0$.

   Since Equation (E.49) is a cubic polynomial equation in β, we can find the roots using the general formula (Spiegel and Liu, 1999) as

   $$\beta_k = -\frac{1}{3a}\left(-N + u_k C + \frac{\Delta_0}{u_k C}\right), \tag{E.59}$$

   where

   $$u_1 = 1, \qquad u_2 = \frac{-1 + j\sqrt{3}}{2}, \qquad u_3 = \frac{-1 - j\sqrt{3}}{2}, \tag{E.60}$$

and

$$C = \sqrt[3]{\frac{\Delta_1 + \sqrt{\Delta_1^2 - 4\Delta_0^3}}{2}}, \tag{E.61}$$

with

$$\begin{aligned}\Delta_0 &= N^2 \\ \Delta_1 &= -2N^3 - 108a^2\kappa\left\|\left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1}\right\|^2.\end{aligned} \tag{E.62}$$

The discriminant for this polynomial equation is given as

$$\Delta = -16\kappa\left\|\left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1}\right\|^2\left(N^3 + 27a^2\kappa\left\|\left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1}\right\|^2\right) < 0, \tag{E.63}$$

which means that there is only one real solution (Spiegel and Liu, 1999). Since β is the precision of the noise $\boldsymbol{\epsilon}$, it must be real valued and positive. Therefore, the only meaningful solution for β is taking u_1, which results in

$$\beta = \frac{1}{3a}\left(N + \frac{N^2+1}{\sqrt[3]{N^3 + 54a^2\kappa\left\|\left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1}\right\|^2 + 6\sqrt{3}\sqrt{a^2N^3\kappa\left\|\left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1}\right\|^2 + 27a^4\kappa^2\left\|\left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1}\right\|^4}}}\right) \tag{E.64}$$

From Equation (E.50), it follows that

$$\mathbf{m}_0 = \mathbf{m}_N. \tag{E.65}$$

Substituting the above result in Equation (E.51), results in

$$\mathbf{L}\mathbf{L}^{\mathrm{T}} = \mathbf{S}_N + 3\kappa\mathbf{L}\mathbf{L}^{\mathrm{T}}\mathbf{L}\mathbf{L}^{\mathrm{T}}\mathbf{L}\mathbf{L}^{\mathrm{T}}. \tag{E.66}$$

Taking the trace of the above equation we can obtain κ as

$$\begin{aligned}\mathrm{Tr}\left(3\kappa\left(\mathbf{L}\mathbf{L}^{\mathrm{T}}\right)^3\right) &= \mathrm{Tr}\left(\mathbf{L}\mathbf{L}^{\mathrm{T}} - \mathbf{S}_N\right) \\ 3\kappa\,\mathrm{Tr}\left(\left(\mathbf{L}\mathbf{L}^{\mathrm{T}}\right)^3\right) &= \mathrm{Tr}\left(\mathbf{L}\mathbf{L}^{\mathrm{T}} - \mathbf{S}_N\right) \\ \kappa &= \frac{\mathrm{Tr}\left(\mathbf{L}\mathbf{L}^{\mathrm{T}} - \mathbf{S}_N\right)}{3\,\mathrm{Tr}\left(\left(\mathbf{L}\mathbf{L}^{\mathrm{T}}\right)^3\right)}.\end{aligned} \tag{E.67}$$

Using the Woodbury identity[7] we can rewrite $\mathbf{S}_N$ from Equation (E.22) as

$$\begin{aligned}\mathbf{S}_N &= \mathbf{S}_0 - \mathbf{S}_0\mathbf{\Phi}^{\mathrm{T}}\left(\frac{1}{\beta}\mathbf{I} + \mathbf{\Phi}\mathbf{S}_0\mathbf{\Phi}^{\mathrm{T}}\right)^{-1}\mathbf{\Phi}\mathbf{S}_0 \\ &= \mathbf{L}\mathbf{L}^{\mathrm{T}} - \mathbf{L}\mathbf{L}^{\mathrm{T}}\mathbf{\Phi}^{\mathrm{T}}\left(\frac{1}{\beta}\mathbf{I} + \mathbf{\Phi}\mathbf{L}\mathbf{L}^{\mathrm{T}}\mathbf{\Phi}^{\mathrm{T}}\right)^{-1}\mathbf{\Phi}\mathbf{L}\mathbf{L}^{\mathrm{T}}.\end{aligned} \tag{E.68}$$

Substituting the above equation in Equation (E.67) we get

$$\kappa = \frac{\mathrm{Tr}\left(\mathbf{L}\mathbf{L}^{\mathrm{T}}\mathbf{\Phi}^{\mathrm{T}}\left(\frac{1}{\beta}\mathbf{I} + \mathbf{\Phi}\mathbf{L}\mathbf{L}^{\mathrm{T}}\mathbf{\Phi}^{\mathrm{T}}\right)^{-1}\mathbf{\Phi}\mathbf{L}\mathbf{L}^{\mathrm{T}}\right)}{3\,\mathrm{Tr}\left(\left(\mathbf{L}\mathbf{L}^{\mathrm{T}}\right)^3\right)}, \tag{E.69}$$

which is strictly positive, since $\mathbf{L}\mathbf{L}^{\mathrm{T}}\mathbf{\Phi}^{\mathrm{T}}\left(\frac{1}{\beta}\mathbf{I} + \mathbf{\Phi}\mathbf{L}\mathbf{L}^{\mathrm{T}}\mathbf{\Phi}^{\mathrm{T}}\right)^{-1}\mathbf{\Phi}\mathbf{L}\mathbf{L}^{\mathrm{T}}$ is the product of positive definite matrices.

[7] See Equation (H.18) in Appendix H.2

Both solutions satisfy the KKT conditions, nevertheless, the solution for $\kappa > 0$ enforces that the norm of the estimated covariance to attain the norm of the CRLB and bounds the value of β.

Algorithm E.1 summarizes the computation of the parameters of the prior distribution of the weights and the precision of the noise using the EM algorithm described in this appendix.

Algorithm E.1: EM Algorithm for estimating $\mathbf{m}_0$, $\mathbf{S}_0$ and β

Input: $\mathbf{\Phi} \in \mathbb{R}^{N\times K}$, $\mathbf{t} \in \mathbb{R}^{N\times 1}$, $\varepsilon \in \mathbb{R}_{\geq 0}$: tolerance

Initialize $\mathbf{m}_0^{(old)}$, $\mathbf{L}^{(old)} = \text{Cholesky}\left(\mathbf{S}_0^{(old)}\right)$, $\beta^{(old)}$, $\kappa^{(old)}$,
$\mathcal{L}^{(old)} := \mathcal{L}_{em}(\beta^{(old)}, \mathbf{m}_0^{(old)}, \mathbf{L}^{(old)}, \kappa^{(old)})$;

Set $Convergence := False$;

while $Convergence\ is\ False$ **do**

Compute new parameters

$$\mathbf{S}_N := \left(\left(\mathbf{L}^{(old)}\mathbf{L}^{(old)^{\mathrm{T}}}\right)^{-1} + \beta^{(old)}\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1} \tag{E.70}$$

$$\mathbf{m}_N := \mathbf{S}_N\left(\left(\mathbf{L}^{(old)}\mathbf{L}^{(old)^{\mathrm{T}}}\right)^{-1}\mathbf{m}_0^{(old)} + \beta^{(old)}\mathbf{\Phi}^{\mathrm{T}}\mathbf{t}\right) \tag{E.71}$$

$$a := \mathbf{t}^{\mathrm{T}}\mathbf{t} - 2\mathbf{t}^{\mathrm{T}}\mathbf{\Phi}\mathbf{m}_N + \mathrm{Tr}\left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\mathbf{S}_N\right) + \mathbf{m}_N^{\mathrm{T}}\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\mathbf{m}_N \tag{E.72}$$

$$\mathbf{m}_0^{(new)} := \mathbf{m}_N \tag{E.73}$$

$$\mathbf{L}^{(new)} := \text{Cholesky}\left(\mathbf{S}_N + \kappa^{(old)}\left(\mathbf{L}^{(old)}\mathbf{L}^{(old)^{\mathrm{T}}}\right)^3\right) \tag{E.74}$$

$$\beta^{(new)} := \frac{1}{3a}\left(N + \frac{N^2+1}{\sqrt[3]{N^3 + 54a^2\kappa^{(old)}\left\|\left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1}\right\|^2 + 6\sqrt{3}\sqrt{a^2N^3\kappa^{(old)}\left\|\left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1}\right\|^2 + 27a^4\kappa^{(old)^2}\left\|\left(\mathbf{\Phi}^{\mathrm{T}}\mathbf{\Phi}\right)^{-1}\right\|^4}}}\right) \tag{E.75}$$

$$\kappa^{(new)} := \frac{\mathrm{Tr}\left(\mathbf{L}^{(old)}\mathbf{L}^{(old)^{\mathrm{T}}} - \mathbf{S}_N\right)}{3\,\mathrm{Tr}\left(\left(\mathbf{L}\mathbf{L}^{\mathrm{T}}\right)^3\right)} \tag{E.76}$$

$$\mathcal{L}^{(new)} := \mathcal{L}_{em}(\beta^{(new)}, \mathbf{m}_0^{(new)}, \mathbf{S}_0^{(new)}, \kappa^{(new)}) \tag{E.77}$$

if $|\mathcal{L}^{(old)} - \mathcal{L}^{(new)}| \leq \varepsilon$ **then**

Set $Convergence := True$;

else

Update parameters

$$\beta^{(old)} := \beta^{(new)} \qquad \mathbf{m}_0^{(old)} := \mathbf{m}_0^{(new)} \qquad \mathbf{L}^{(old)} := \mathbf{L}^{(new)} \qquad \kappa^{(old)} := \kappa^{(new)} \qquad \mathcal{L}^{(old)} := \mathcal{L}^{(new)} \tag{E.78}$$

return

$$\mathbf{m}_0 = \mathbf{m}_0^{(new)} \qquad \mathbf{S}_0 = \mathbf{L}^{(new)}\mathbf{L}^{(new)^{\mathrm{T}}} \qquad \beta = \beta^{(new)} \tag{E.79}$$

F Building and Training Neural Network Models

This appendix provides some details for training the neural network models described in Chapter 4, as well as details of the recurrent layers used in the experiments described in Chapters 4, 5, 6.

F.1 RMSProp

As described in Section 4.2.2, given the nonlinear activations of neural network models, it is not generally possible to compute a closed form analytical solution to the model parameters $\boldsymbol{\theta}$ that minimize a loss function $\mathcal{L}$ (i.e. $\hat{\boldsymbol{\theta}} = \text{argmin}_{\boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta})$) in the same way as described for the linear models in Appendix E. Therefore, it is necessary to use numerical methods, such as variants of the stochastic gradient descent (SGD) algorithm (Boyd and Vandenberghe, 2004; Nocedal and Wright, 2006; Goodfellow et al., 2016).

In this work, we have made extensive use of RMSProp (Tieleman and Hinton, 2012), a variant of SGD that adaptively change the value of the learning rate In this section we describe how to compute the parameter updates using RMSProp.

The learning rate for update q (denoted as η_q) is adaptively computed as

$$\eta_q = \frac{\eta}{\sqrt{r_q + \varepsilon}} \tag{F.1}$$

where r_q is a moving average estimation of the norm of the gradient of the loss function with respect to the parameters, η is the initial learning rate, and ε is a small value (a machine epsilon) added for numerical stability. In all experiments reported on this thesis, we use $\varepsilon = 10^{-6}$. The moving average estimation of the norm of the gradient of the loss function is computed as

$$r_q = (1-\gamma)\left\| \frac{\partial}{\partial \boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_q} \right\|^2 + \gamma \cdot r_{q-1}, \tag{F.2}$$

where $\gamma \in [0, 1]$ is a decay factor for the computation of the moving average. A value of γ close to 1 will decay the moving average slowly, and a value close to 0 will decay the moving average fast. Finally, the updates of the parameters are computed as

$$\boldsymbol{\theta}_{q+1} = \boldsymbol{\theta}_q - \eta_q \frac{\partial}{\partial \boldsymbol{\theta}} \mathcal{L}(\boldsymbol{\theta}) \Big|_{\boldsymbol{\theta}=\boldsymbol{\theta}_q}. \tag{F.3}$$

F.2 Recurrent Layers Used in this Work

In this section, we describe the recurrent layers used for the experiments described in the main text.

F.2.1 Vanilla Recurrent Layer

Let the l-th layer of a neural network be a vanilla recurrent layer. The output of such network for the i-th time step of an input sequence is given as

$$\boldsymbol{h}_i^{(l)} = \sigma^{(l)}\left(\mathbf{w}_x^{(l)}\boldsymbol{h}_i^{(l-1)} + \mathbf{w}_h^{(l)}\boldsymbol{h}_{i-1}^{(l)} + \mathbf{w}_0^{(l)}\right). \tag{F.4}$$

where $\mathbf{w}_x^{(l)} \in \mathbb{R}^{D_l \times D_{l-1}}$ is a matrix of weights connecting the $(l-1)$-th layer $\boldsymbol{h}_i^{(l)}$, $\mathbf{w}_h^{(l)} \in \mathbb{R}^{D_l \times D_{l-1}}$ is a matrix of weights connecting the $(i-1)$-th activation of $\boldsymbol{h}^{(l)}$; and $\mathbf{w}_0^{(l)} \in \mathbb{R}^{D_l}$ are the matrix of weights and the bias vector of the l-th layer. D_l is the number of units of layer $\boldsymbol{h}^{(l)}$.

While it is theoretically possible for large enough vanilla RNNs to predict sequences of arbitrary complexity, it has been shown that numerical limitations of training algorithms do not allow them to properly model long temporal dependencies (Pascanu et al., 2013).

F.2.2 LSTM with Multiplicative Integration

Let the l-th layer of a neural network be a long short term memory layer (LSTM) with multiplicative integration (Wu et al., 2016). The output of such layer for the i-th time step of an input sequence is given as

$$\boldsymbol{z}_i^{(l)} = \tanh(\boldsymbol{\beta}_{z,1}^{(l)} \odot \mathbf{U}_z^{(l)}\boldsymbol{h}_{i-1}^{(l)} + \boldsymbol{\beta}_{z,2}^{(l)} \odot \mathbf{w}_z^{(l)}\boldsymbol{h}_i^{(l-1)} + \boldsymbol{\alpha}_z^{(l)} \odot \mathbf{w}_z^{(l)}\boldsymbol{h}_i^{(l-1)} \odot \mathbf{U}_z^{(l)}\boldsymbol{h}_{i-1}^{(l)} + \mathbf{b}_z^{(l)}) \tag{F.5}$$

$$\boldsymbol{j}_i^{(l)} = \sigma(\boldsymbol{\beta}_{j,1}^{(l)} \odot \mathbf{U}_j^{(l)}\boldsymbol{h}_{i-1}^{(l)} + \boldsymbol{\beta}_{j,2}^{(l)} \odot \mathbf{w}_j^{(l)}\boldsymbol{h}_i^{(l-1)} + \boldsymbol{\alpha}_i^{(l)} \odot \mathbf{w}_i^{(l)}\boldsymbol{h}_i^{(l-1)} \odot \mathbf{U}_j^{(l)}\boldsymbol{h}_{i-1}^{(l)} + \mathbf{b}_j^{(l)}) \tag{F.6}$$

$$\boldsymbol{f}_i^{(l)} = \sigma(\boldsymbol{\beta}_{f,1}^{(l)} \odot \mathbf{U}_f^{(l)}\boldsymbol{h}_{i-1}^{(l)} + \boldsymbol{\beta}_{f,2}^{(l)} \odot \mathbf{w}_f^{(l)}\boldsymbol{h}_i^{(l-1)} + \boldsymbol{\alpha}_f^{(l)} \odot \mathbf{w}_f^{(l)}\boldsymbol{h}_i^{(l-1)} \odot \mathbf{U}_f^{(l)}\boldsymbol{h}_{i-1}^{(l)} + \mathbf{b}_f^{(l)}) \tag{F.7}$$

$$\boldsymbol{o}_i^{(l)} = \sigma(\boldsymbol{\beta}_{o,1}^{(l)} \odot \mathbf{U}_o^{(l)}\boldsymbol{h}_{i-1}^{(l)} + \boldsymbol{\beta}_{o,2}^{(l)} \odot \mathbf{w}_o^{(l)}\boldsymbol{h}_i^{(l-1)} + \boldsymbol{\alpha}_o^{(l)} \odot \mathbf{w}_o^{(l)}\boldsymbol{h}_i^{(l-1)} \odot \mathbf{U}_o^{(l)}\boldsymbol{h}_{i-1}^{(l)} + \mathbf{b}_o^{(l)}) \tag{F.8}$$

$$\boldsymbol{c}_i^{(l)} = \boldsymbol{j}_i^{(l)} \odot \boldsymbol{z}_i^{(l)} + \boldsymbol{f}_i^{(l)} \odot \boldsymbol{c}_{i-1}^{(l)} \tag{F.9}$$

$$\boldsymbol{h}_i^{(l)} = \boldsymbol{o}_i^{(l)} \odot \tanh(\boldsymbol{c}_i{}^{(l)}), \tag{F.10}$$

where $\odot$ is the Hadamard product (i.e. the element-wise product), $\boldsymbol{z}_i^{(l)}, \boldsymbol{h}_i^{(l)} \in \mathbb{R}^{D_l}$ are the l-th *block input* and l-th *output*, respectively, $\boldsymbol{c}_i^{(l)} \in \mathbb{R}^{D_l}$ is the l-th *cell state* and $\boldsymbol{j}_i^{(l)}, \boldsymbol{f}_i^{(l)}, \boldsymbol{o}_i^{(l)} \in \mathbb{R}^{D_l}$ are the l-th *input*, *forget* and *output gates*, respectively. The parameters of an MI-LSTM layer are $\{\mathbf{w}_k^{(l)} \mid k \in \{z, j, f, o\}\}$, input-to-hidden weight matrices in $\mathbb{R}^{D_l \times D_{l-1}}$, $\{\mathbf{U}_k^{(l)} \mid k \in \{z, j, f, o\}\}$, hidden-to-hidden weight matrices in $\mathbb{R}^{D_l \times D_l}$, $\{\mathbf{b}_k^{(l)} \mid k \in \{z, j, f, o\}\}$, bias vectors in $\mathbb{R}^{D_l}$ and $\{\boldsymbol{\alpha_k}^{(l)}, \boldsymbol{\beta}_{k,1}^{(l)}, \boldsymbol{\beta}_{k,2}^{(l)} \mid k \in \{z, j, f, o\}\}$, gating bias vectors for multiplicative integration in $\mathbb{R}^{D_l}$. $\sigma(\cdot)$ and $\tanh(\cdot)$ are the elementwise sigmoid and hyperbolic tangent functions, respectively.

G Sampling from Generative Probabilistic Models

G.1 Introduction

In this appendix, we describe methods for generating samples from the probabilistic BMs described in the main text.

In Chapter 3 we developed a probabilistic extension of the linear BMs. Given a model trained over a training set $\boldsymbol{\mathcal{T}}$, we can make predictions for a new score $\mathfrak{S}$ represented by $\boldsymbol{\Phi}$. These predictions can be expressed in terms of the predictive distribution over $\mathbf{T}$, the value of the expressive parameters, rather than simply a point estimate. The predictive distribution for $t_{ij} \in \mathbf{T}$, the j-th expressive parameter for the i-th element in the score is obtained by marginalizing with respect to the posterior distribution of the parameters of the predictive function $\boldsymbol{\theta}$, i.e.

$$p(t_{ij} \mid \boldsymbol{\varphi}(x_i), \boldsymbol{\mathcal{T}}) = \int p(t_{ij} \mid \boldsymbol{\varphi}(x_i), \boldsymbol{\theta}_j) p(\boldsymbol{\theta} \mid \boldsymbol{\mathcal{T}}) d\boldsymbol{\theta}_j, \tag{G.1}$$

where $\boldsymbol{\theta}_j$ are the parameters for the predictive function for the j-th parameter; $p(t_{ij} \mid \boldsymbol{\varphi}(x_i), \boldsymbol{\theta}_j)$ is the conditional probability of the value of the expressive parameters given the model parameters and the input basis functions; and $p(\boldsymbol{\theta}_j \mid \boldsymbol{\mathcal{T}})$ is the conditional distribution of the model parameters given the training set.

We model conditional probability of the value of the expressive parameter t_{ij} given the model parameters and the input basis functions as a Gaussian distribution with precision β_j^{-1} centered around the output of the predictive function $f^{(j)}(\boldsymbol{\varphi}(x_i)\,;\boldsymbol{\theta}_j)$, i.e.,

$$p(t_{ij} \mid \boldsymbol{\Phi}, \boldsymbol{\theta}_j) = \mathcal{N}(t_{ij} \mid f^{(j)}(\boldsymbol{\varphi}(x_i)\,;\boldsymbol{\theta}_j), \beta_j^{-1}). \tag{G.2}$$

Using Bayes' theorem, we can compute the conditional distribution $p(\boldsymbol{\theta}_j \mid \boldsymbol{\mathcal{T}})$ as

$$p(\boldsymbol{\theta}_j \mid \boldsymbol{\mathcal{T}}) = \frac{p(\boldsymbol{\mathcal{T}} \mid \boldsymbol{\theta}_j) p(\boldsymbol{\theta}_j)}{\int p(\boldsymbol{\mathcal{T}} \mid \boldsymbol{\theta}_j) p(\boldsymbol{\theta}_j) d\boldsymbol{\theta}_j}, \tag{G.3}$$

where $p(\boldsymbol{\theta}_j)$ is the prior distribution of the parameters of the model for the j-th expressive parameter.

To unclutter notation, in the following discussion we will write $\boldsymbol{\varphi}(x_i) = \boldsymbol{\varphi}_i$ and drop the subscript j in all equations, since the derivation of the predictive distribution has the same form for all expressive parameters.

G.2 Linear Basis Models

We can write the predictive distribution for the i-th expressive target as follows. Due to the Gaussian assumptions on the prior distribution of the parameters (i.e. $p(\boldsymbol{\theta}) = \mathcal{N}(\boldsymbol{\theta} \mid \mathbf{m}_0, \mathbf{S}_0)$,

and the linear dependency of the model parameters $\boldsymbol{\theta}$, the posterior probability $p(\boldsymbol{\theta} \mid \boldsymbol{\mathcal{T}})$ is also Gaussian given by[1]

$$p(\boldsymbol{\theta} \mid \boldsymbol{\Phi}, \boldsymbol{\mathcal{T}}) = \mathcal{N}(\boldsymbol{\theta} \mid \mathbf{m}_N, \mathbf{S}_N), \tag{G.4}$$

where $\mathbf{m}_N$ is the posterior mean and $\mathbf{S}_N$ is the posterior covariance matrix, calculated using Equation (3.29).

Substituting Equations (3.26) and (G.4) in Equation (G.1) results in the marginalization of a Gaussian distribution, which is also Gaussian[2]. Therefore, we can write the predictive distribution for the value of the expressive parameter for the i-th element in the score as

$$p(t_i \mid \boldsymbol{\Phi}, \boldsymbol{\mathcal{T}}) = \mathcal{N}(t_i \mid f_{lbm}(\boldsymbol{\varphi}_i\,; \bar{\boldsymbol{\theta}}), \sigma_N^2), \tag{G.5}$$

where f_{lbm} is the predictive function, $\bar{\boldsymbol{\theta}}$ are the parameters of this predictive function trained on training set $\boldsymbol{\mathcal{T}}$[3], and σ^2 is the variance of the distribution, given by

$$\sigma_N^2 = \frac{1}{\beta} + \boldsymbol{\varphi}_i^{\mathrm{T}} \left(\mathbf{S}_0^{-1} + \beta \boldsymbol{\Phi}_{\boldsymbol{\mathcal{T}}}^{\mathrm{T}} \boldsymbol{\Phi}_{\boldsymbol{\mathcal{T}}}\right)^{-1} \boldsymbol{\varphi}_i. \tag{G.6}$$

where $\boldsymbol{\Phi}_{\boldsymbol{\mathcal{T}}}$ is the matrix concatenating all score representations in the training set, given using Equation (3.22).

G.3 (Recurrent) Non-linear Basis Models

For the both the FFNN-based non-sequential non-linear BM (NBM) described in Section 4.2 and the RNN-based sequential BM described in Section 4.3 (RNBM), the integral in Equation (G.1) is analytically intractable, due to the posterior probability of the parameters being non-Gaussian, and due to the non-linear dependencies of the parameters in the conditional distribution of the target variables. In order to solve this problem, we can take two alternatives:

1. *Linear-Gaussian approximation.* We can use the Laplace approximation (Bishop, 1995) to find a Gaussian approximation to the posterior distribution as

   $$p(\boldsymbol{\theta} \mid \boldsymbol{\mathcal{T}}) \approx q(\boldsymbol{\theta} \mid \boldsymbol{\mathcal{T}}) = \mathcal{N}\left(\boldsymbol{\theta} \mid \bar{\boldsymbol{\theta}}, \mathbf{A}^{-1}\right), \tag{G.7}$$

   where $\bar{\boldsymbol{\theta}}$ are the parameters of the trained model and $\mathbf{A}$ is the matrix of second derivatives of the negative log posterior distribution, given by

   $$\mathbf{A} = -\frac{\partial^2}{\partial \boldsymbol{\theta}^2} \log p(\boldsymbol{\theta} \mid \boldsymbol{\mathcal{T}}) = \alpha \mathbf{I} + \beta \mathbf{R}, \tag{G.8}$$

   where $\mathbf{R}$ is the Hessian matrix of the squared error with respect to $\boldsymbol{\theta}$, i.e.

   $$\mathbf{R} = \frac{\partial^2}{\partial \boldsymbol{\theta}^2} \| f_{(r)nbm}(\boldsymbol{\varphi}_i\,; \boldsymbol{\theta}) - t_i \|^2, \tag{G.9}$$

   where $f_{(r)nbm}(\boldsymbol{\varphi}_i\,; \boldsymbol{\theta})$ is the output of the predictive function (since this discussion applies to both NBM and RNBM, $f_{(r)nbm}$ denotes the output of either model). We can further approximate the inverse of $\mathbf{A}$ as[4]

   $$\mathbf{A}^{-1} \approx \frac{1}{\alpha}\mathbf{I} - \frac{\beta}{\alpha^2}\mathbf{R}. \tag{G.10}$$

[1] For a detailed derivation see Appendix E.3.
[2] See Section 2.3.3 in (Bishop, 2006) for a detailed derivation of this result.
[3] See Section 3.3.3 for a discussion of the training of the model
[4] This approximation holds if $\beta \ll \alpha$. See (Petersen and Pedersen, 2012).

Algorithm G.1: Approximation of the predictive distribution using importance sampling.

Input:

- $\boldsymbol{\mathcal{T}}$: training set
- $\mathfrak{S}$: score to be rendered
- $q(\boldsymbol{\theta} \mid \mathfrak{S}, \boldsymbol{\mathcal{T}})$: proposal posterior distribution

1 Generate a set of L samples from the proposal distribution as

$$\boldsymbol{\theta}^* = \{\boldsymbol{\theta}_1^*, \ldots, \boldsymbol{\theta}_L^* \mid \boldsymbol{\theta}_i^* \sim q(\boldsymbol{\theta} \mid \boldsymbol{\mathcal{T}}) \; \forall i \in [1, L]\} \tag{G.15}$$

2 Compute the set of importance weights as

$$\mathbf{r} = \left\{ r_1, \ldots, r_L \mid r_i = \frac{\tilde{p}(\boldsymbol{\theta}_i^* \mid \boldsymbol{\mathcal{T}})}{\tilde{q}(\boldsymbol{\theta}_i^* \mid \boldsymbol{\mathcal{T}})}, \; \forall \boldsymbol{\theta}_i^* \in \boldsymbol{\theta}^* \right\} \tag{G.16}$$

3 Compute the normalized importance weights as

$$\boldsymbol{w} = \left\{ w_1, \ldots, w_L \mid w_i = \frac{r_l}{\sum_{l=1}^{L} r_l}, \; \forall r_i \in \mathbf{r} \right\} \tag{G.17}$$

4 **return** Predictive distribution

$$p(\mathbf{T} \mid \boldsymbol{\mathcal{T}}) \approx \sum_{l=1}^{L} w_l p(\mathbf{T} \mid \boldsymbol{\theta}_l^*) \tag{G.18}$$

Using a Taylor expansion of the model output around the $\bar{\boldsymbol{\theta}}$

$$f_{(r)nbm}(\boldsymbol{\varphi}_i, \boldsymbol{\theta}) \approx \mu_{lin} = f_{(r)nbm}(\boldsymbol{\varphi}_i \,; \bar{\boldsymbol{\theta}}) + (\boldsymbol{\theta} - \bar{\boldsymbol{\theta}})^{\mathrm{T}} \left. \frac{\partial}{\partial \boldsymbol{\theta}} f(\boldsymbol{\varphi}_i \,; \boldsymbol{\theta}) \right|_{\boldsymbol{\theta}=\bar{\boldsymbol{\theta}}}, \tag{G.11}$$

We can then write an approximation of the conditional probability $p(t_i \mid \boldsymbol{\theta})$ that is Gaussian, and whose mean is a linear function of $\boldsymbol{\theta}$ as

$$p(t_i \mid \boldsymbol{\theta}) \approx \mathcal{N}(t_i \mid \mu_{lin}, \beta^{-1}). \tag{G.12}$$

These two approximations allow us to approximate the integral in Equation (G.1) in a similar way as in the linear case as

$$p(t_i \mid \boldsymbol{\mathcal{T}}) = \mathcal{N}\left(t_i \mid (\boldsymbol{\varphi}_i \,; \bar{\boldsymbol{\theta}}), \sigma_N^2\right), \tag{G.13}$$

with

$$\sigma_N^2 = \frac{1}{\beta} + \left(\left. \frac{\partial}{\partial \boldsymbol{\theta}} f(\boldsymbol{\varphi}_i \,; \boldsymbol{\theta}) \right|_{\boldsymbol{\theta}=\bar{\boldsymbol{\theta}}} \right)^{\mathrm{T}} \left(\frac{1}{\alpha} \mathbf{I} - \frac{\beta}{\alpha^2} \, \mathbf{R}|_{\boldsymbol{\theta}=\bar{\boldsymbol{\theta}}} \right) \left(\left. \frac{\partial}{\partial \boldsymbol{\theta}} f(\boldsymbol{\varphi}_i \,; \boldsymbol{\theta}) \right|_{\boldsymbol{\theta}=\bar{\boldsymbol{\theta}}} \right). \tag{G.14}$$

From the above results it follows that the linear-Gaussian approximation to compute the predictive distribution leads to a unimodal distribution.

2. *Use Monte Carlo integration.*

We can reformulate the computation of the predictive distribution from Equation (G.1) as calculating the expectation value of the conditional probability of $\mathbf{T}$ given the parameters

$\boldsymbol{\theta}$ as

$$p(t_i \mid \boldsymbol{\mathcal{T}}) = \mathbb{E}\left\{p(t_i \mid \boldsymbol{\theta})\right\} \tag{G.19}$$

We can use *importance sampling*, a Monte Carlo technique for approximating expectation values (Bishop, 2006). The posterior distribution $p(\boldsymbol{\theta} \mid \boldsymbol{\mathcal{T}})$ can be rewritten as

$$p(\boldsymbol{\theta} \mid \boldsymbol{\mathcal{T}}) = \frac{\overbrace{\tilde{p}(\boldsymbol{\theta} \mid \boldsymbol{\mathcal{T}})}^{p(\boldsymbol{\mathcal{T}}|\boldsymbol{\theta})p(\boldsymbol{\theta})}}{\underbrace{Z_p}_{\int p(\boldsymbol{\mathcal{T}}|\boldsymbol{\theta})p(\boldsymbol{\theta})d\boldsymbol{\theta}}}, \tag{G.20}$$

where Z_p is a normalization constant known as the *partition function*. Since generating samples from $p(\boldsymbol{\theta} \mid \boldsymbol{\mathcal{T}})$ is complicated, but evaluating $\tilde{p}(\boldsymbol{\theta} \mid \boldsymbol{\mathcal{T}})$ is straightforward, we introduce a *proposal distribution* $q(\boldsymbol{\theta} \mid \boldsymbol{\mathcal{T}})$ from which it is easy to draw samples. A candidate distribution for $q(\boldsymbol{\theta} \mid \boldsymbol{\mathcal{T}})$ is given by the Laplace approximation in Equation (G.7). Furthermore, we can rewrite this proposal distribution to match the form of Equation (G.20) as

$$q(\boldsymbol{\theta} \mid \boldsymbol{\mathcal{T}}) = \frac{\tilde{q}(\boldsymbol{\theta} \mid \boldsymbol{\mathcal{T}})}{Z_q}. \tag{G.21}$$

Using this, we can rewrite the expectation in Equation (G.19) as

$$\begin{aligned}\mathbb{E}\left\{p(t_i \mid \boldsymbol{\theta})\right\} &= \int p(t_i \mid \boldsymbol{\theta}) p(\boldsymbol{\theta} \mid \boldsymbol{\mathcal{T}}) d\boldsymbol{\theta} \\ &= \frac{Z_q}{Z_p} \int p(t_i \mid \boldsymbol{\theta}) \frac{\tilde{p}(\boldsymbol{\theta} \mid \boldsymbol{\mathcal{T}})}{\tilde{q}(\boldsymbol{\theta} \mid \boldsymbol{\mathcal{T}})} q(\boldsymbol{\theta} \mid \boldsymbol{\mathcal{T}}) d\boldsymbol{\theta}\end{aligned} \tag{G.22}$$

By taking L samples from $\boldsymbol{\theta}^* \sim q(\boldsymbol{\theta} \mid \boldsymbol{\mathcal{T}})$, we can approximate the above expectation as

$$\mathbb{E}\left\{p(\mathbf{T} \mid \boldsymbol{\theta})\right\} \approx \frac{Z_q}{Z_p} \frac{1}{L} \sum_{l=1}^{L} r_l p(\mathbf{T} \mid \boldsymbol{\theta}_l^*), \tag{G.23}$$

where r_l is the l-th *importance weight*, given by

$$r_l = \frac{\tilde{p}(\boldsymbol{\theta}_l^* \mid \boldsymbol{\mathcal{T}})}{\tilde{q}(\boldsymbol{\theta}_l^* \mid \boldsymbol{\mathcal{T}})}. \tag{G.24}$$

Finally, we can approximate the ratio of partition functions $\frac{Z_p}{Z_q}$ as

$$\frac{Z_p}{Z_q} \approx \frac{1}{L} \sum_{l=1}^{L} r_l. \tag{G.25}$$

The computation of the predictive distribution using importance sampling is summarized in Algorithm G.1.

H Mathematical Formulae and Identities

This appendix provides a brief summary of formulae and identities used in this work.

H.1 Probability Theory

The probability density function (pdf) of a D-dimensional multivariate Gaussian distribution is given by

$$\mathcal{N}(\mathbf{x} \mid \mathbf{m}, \mathbf{S}) = \frac{1}{(2\pi)^D} \frac{1}{\sqrt{\det(\mathbf{S})}} \exp\left\{-\frac{1}{2}(\mathbf{x}-\mathbf{m})^T \mathbf{S}^{-1}(\mathbf{x}-\mathbf{m})\right\}, \tag{H.1}$$

where $\mathbf{x} \in \mathbb{R}^D$ is a random vector, $\mathbf{m} \in \mathbb{R}^D$ is the mean vector and $\mathbf{S} \in \mathbb{R}^{D \times D}$ is the covariance matrix.

Given a conditional distribution for $\mathbf{y}$ given $\mathbf{x}$ and a marginal distribution for $\mathbf{x}$ in the form

$$p(\mathbf{y} \mid \mathbf{x}) = \mathcal{N}\left(\mathbf{y} \mid \mathbf{A}\mathbf{x} + \mathbf{b}, \mathbf{L}^{-1}\right) \tag{H.2}$$

$$p(\mathbf{x}) = \mathcal{N}\left(\mathbf{x} \mid \mathbf{m}, \mathbf{\Lambda}^{-1}\right), \tag{H.3}$$

the conditional distribution of $\mathbf{x}$ given $\mathbf{y}$ and the marginal distribution of $\mathbf{y}$ are given by

$$p(\mathbf{x} \mid \mathbf{y}) = \mathcal{N}\left(\mathbf{x} \mid \mathbf{\Sigma}\left(\mathbf{A}^T\mathbf{L}(\mathbf{y}-\mathbf{b}) + \mathbf{\Lambda}\mathbf{m}\right), \mathbf{\Sigma}\right) \tag{H.4}$$

$$p(\mathbf{y}) = \mathcal{N}\left(\mathbf{y} \mid \mathbf{A}\mathbf{m} + \mathbf{b}, \mathbf{L}^{-1} + \mathbf{A}\mathbf{\Lambda}^{-1}\mathbf{A}^T\right) \tag{H.5}$$

where

$$\mathbf{\Sigma} = \left(\mathbf{\Lambda} + \mathbf{A}^T\mathbf{L}\mathbf{A}\right)^{-1}. \tag{H.6}$$

For a detailed derivation of this result, see (Bishop, 2006, pp. 90-93).

H.2 Linear Algebra

For a more comprehensive list of linear algebra identities, we refer the reader to (Petersen and Pedersen, 2012) and (Spiegel and Liu, 1999).

- Transposition.

$$(\mathbf{A}+\mathbf{B})^T = \mathbf{A}^T + \mathbf{B}^T \tag{H.7}$$

$$(\mathbf{A}\mathbf{B})^T = \mathbf{B}^T\mathbf{A}^T \tag{H.8}$$

$$\mathbf{A} = \mathbf{A}^T \quad \text{if } \mathbf{A} \text{ is symmetric} \tag{H.9}$$

- Traces.

$$\mathrm{Tr}(\mathbf{A}+\mathbf{B}) = \mathrm{Tr}(\mathbf{A}) + \mathrm{Tr}(\mathbf{B}) \tag{H.10}$$
$$\mathrm{Tr}(\mathbf{ABC}) = \mathrm{Tr}(\mathbf{BCA}) = \mathrm{Tr}(\mathbf{CAB}) \tag{H.11}$$
$$\mathrm{Tr}(a) = a \;\forall a \in \mathbb{C} \tag{H.12}$$
$$\mathrm{Tr}(\mathbf{A}^T) = \mathrm{Tr}(\mathbf{A}) \tag{H.13}$$

- Determinant.

$$\det(\mathbf{AB}) = \det(\mathbf{A})\det(\mathbf{B}) \tag{H.14}$$
$$\det(\mathbf{A}^{-1}) = \frac{1}{\det(\mathbf{A})} \tag{H.15}$$
$$\det(\mathbf{A}^T) = \det(\mathbf{A}) \tag{H.16}$$

- The Frobenius norm.

$$\|\mathbf{A}\|^2 = \mathrm{Tr}\left(\mathbf{AA}^T\right) \tag{H.17}$$

- The Woodbury identity.

$$\left(\mathbf{A}+\mathbf{CBC}^T\right)^{-1} = \mathbf{A}^{-1} - \mathbf{A}^{-1}\mathbf{C}\left(\mathbf{B}^{-1}+\mathbf{C}^T\mathbf{A}^{-1}\mathbf{C}\right)^{-1}\mathbf{C}^T\mathbf{A}^{-1} \tag{H.18}$$

H.3 Miscellaneous

1. Iverson Bracket/ Indicator function.

$$\mathbb{1}\{A\} = \begin{cases} 1 & \text{if } A \text{ is true} \\ 0 & \text{otherwise} \end{cases} \tag{H.19}$$

   Although this notation is more common to the normal definition of the indicator function, we use the formulation of the Iverson bracket, which explicitly converts a logical proposition into a binary representation.

2. Hyberbolic tangent

$$\tanh(x) = \frac{\exp(x)-\exp(-x)}{\exp(x)+\exp(-x)} \tag{H.20}$$

   The hyperbolic tangent is illustrated in Figure H.1 (in red).

3. Sigmoid

$$\sigma(x) = \frac{\exp(x)}{\exp(x)+1} \tag{H.21}$$

   The sigmoid function is illustrated in Figure H.1 (in blue).

4. Rectifier

$$\mathrm{rectifier}(x) = \begin{cases} x & \text{if } x \geq 0 \\ 0 & \text{otherwise} \end{cases} \tag{H.22}$$

   The rectifier function is illustrated in Figure H.1 (in green).

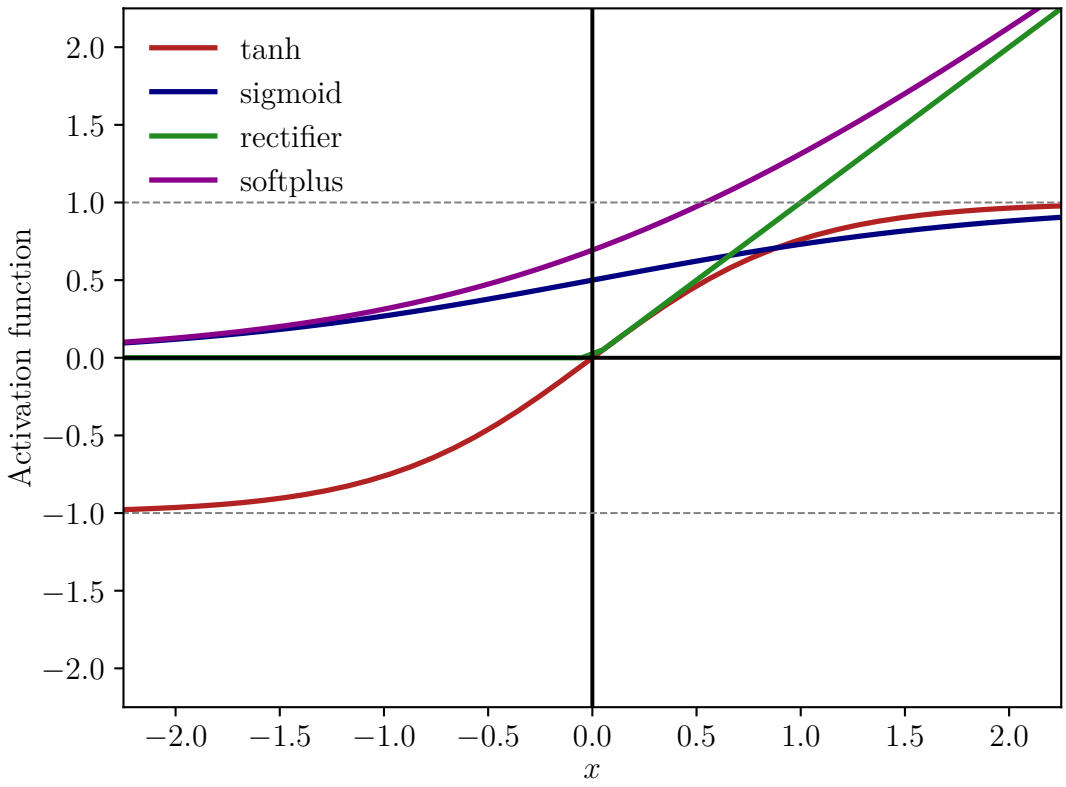


Figure H.1: Comparison of the activation functions used in the neural networks described in the main text.

5. Softplus

$$\text{softplus}(x) = \log\left(1 + \exp(x)\right) \tag{H.23}$$

The softplus function can be thought as a smooth approximation to the rectifier. This function is illustrated in Figure H.1 (in purple).

Curriculum Vitae of the Author

Personal Information

Name	Carlos Eduardo Cancino Chacón
Birth	June 6, 1986
Citizenship	Mexican
Address	Austrian Research Institute for Artificial Intelligence Freyung 6 / 6 1010, Vienna, Austria
E-Mail	carlos.cancino@ofai.at
Website	http://www.carloscancinochacon.com

Academic Qualifications

PhD in Computer Science 10/2014–present

Johannes Kepler University of Linz, Austria

Supervisor: Gerhard Widmer
Thesis: Computational Modeling of Expressive Music Performance with Linear and Non-linear Basis Function Models.

M.Sc. in Electrical Engineering and Audio Engineering 10/2011–07/2014

Graz University of Technology/University of Music and Performing Arts Graz, Austria

Supervisor: Franz Pernkopf
Thesis: Tarkus Belief Propagation: On Message Passing Algorithms and Computational Commutative Algebra.
With distinction.

Licenciatura como Físico 08/2005–03/2011

National Autonomous University of Mexico, Mexico City, Mexico

Supervisor: Marcos Ley Koo
Thesis: Análisis teórico experimental de transductores de ultrasonido tipo Langevin.

Licenciatura en Concertista de Piano 09/1999–02/2011

National Conservatory of Music, Mexico City, Mexico

Supervisor: Héctor Alfonso Rojas Ramírez

Research Experience

Researcher 01/2014–present

Austrian Research Institute for Artificial Intelligence
Intelligent Music Processing and Machine Learning Group

Teaching

Course Lecturer Level B 01/2011–07/2011

National Conservatory of Music, Mexico City, Mexico

Courses: Elementary Music Theory I and Harmony (Levels I-III)

Academic Activities

INVITED TALKS

1. *¿Escuchan los androides música electrónica?* (2016). Talk series: Pláticas DeMentes. Faculty of Psychology, National Autonomous University of Mexico.
2. *En busca del factor Mozart* (2016). National Conservatory of Music, Mexico City.

REVIEWING

1. **Journals:** Journal of New Music Research (2017), Neural Computing and Applications (2018)
2. **Conferences:** MCM (2015), DLM17 (2017), ISMIR (2014, 2015, 2016, 2017, 2018)

Awards and Grants

Fundación INBA – CONACYT Scholarship
Mexican National Council for Science and Technology 10/2012-02/2014

€13,355 per annum stipend, tuition fees and medical insurance to study at Graz University of Technology

Award for Creative Achievement 06/2017

AccompaniX Competition, 2017 Turing Tests in the Creative Arts.
$500 team award for development of an expressive computer accompaniment system.

Publications

PEER REVIEWED PUBLICATIONS

1. Cancino-Chacón, C., Grachten, M., Goebl, W., and Widmer, G. (2018). Computational Models of Expressive Music Performance: A Comprehensive and Critical Review. *To appear in Frontiers in Digital Humanities*

2. Cancino-Chacón, C., Grachten, M., Sears, D. R. W., and Widmer, G. (2017c). What Were You Expecting? Using Expectancy Features to Predict Expressive Performances of Classical Piano Music. In *Proceedings of the 10th International Workshop on Machine Learning and Music (MML 2017)*, Barcelona, Spain

3. Cancino-Chacón, C., Grachten, M., and Agres, K. (2017b). From Bach to The Beatles: The Simulation of Human Tonal Expectation Using Ecologically-Trained Predictive Models. In *Proceedings of the 18th International Society for Music Information Retrieval Conference (ISMIR 2017)*, Suzhou, China

4. Cancino-Chacón, C. E., Gadermaier, T., Widmer, G., and Grachten, M. (2017d). An Evaluation of Linear and Non-linear Models of Expressive Dynamics in Classical Piano and Symphonic Music. *Machine Learning*, 106(6):887–909

5. Grachten, M., Cancino-Chacón, C. E., Gadermaier, T., and Widmer, G. (2017). Towards computer-assisted understanding of dynamics in symphonic music. *IEEE Multimedia*, 24(1):36–46

6. Grachten, M. and Cancino-Chacón, C. E. (2017). Temporal dependencies in the expressive timing of classical piano performances. In Lessafre, M., Maes, P.-J., and Leman, M., editors, *The Routledge Companion to Embodied Music Interaction*, pages 360–369. Routledge

7. Velarde, Gissel and Weyde, Tillman and Cancino Chacón, Carlos and Meredith, David and Grachten, Maarten (2016). Composer Recognition Based On 2D-Filtered Piano Rolls. In *Proceedings of the 17th International Society for Music Information Retrieval Conference (ISMIR 2016)*, pages 116–121, New York, NY, USA

8. Gadermaier, T., Grachten, M., and Cancino-Chacón, C. E. (2016). Basis-Function Modeling of Loudness Variations in Ensemble Performance. In *Proceedings of the 2nd International Conference on New Music Concepts (ICNMC 2016)*, Treviso, Italy

9. Cancino Chacón, C. E. and Grachten, M. (2015). An Evaluation of Score Descriptors Combined with Non-linear Models of Expressive Dynamics in Music. In *Proceedings of the 18th International Conference on Discovery Science (DS 2015)*, pages 48–62, Banff, AB, Canada

10. Agres, K., Cancino, C., Grachten, M., and Lattner, S. (2015). Harmonics co-occurrences bootstrap pitch and tonality perception in music: Evidence from a statistical unsupervised learning model. In *Proceedings of the Annual Meeting of the Cognitive Science Society (CogSci 2015)*, Pasadena, CA, USA

11. Lattner, S., Grachten, M., Agres, K., and Cancino Chacón, C. E. (2015b). Probabilistic Segmentation of Musical Sequences using Restricted Boltzmann Machines. In *Fifth International Conference on Mathematics and Computation in Music (MCM 2015)*, London, UK

12. Lattner, S., Cancino Chacón, C. E., and Grachten, M. (2015a). Pseudo-Supervised Training Improves Unsupervised Melody Segmentation. In *In Proceedings of the Twenty-Fourth*

International Joint Conference on Artificial Intelligence (IJCAI 2015), pages 2459–2465, Buenos Aires, Argentina

13. Cancino Chacón, C. E. and Mowlaee, P. (2014). Least Squares Phase Estimation of Mixed Signals. In *15th Annual Conference of the International Speech Communication Association (INTERSPEECH 2014)*, Singapore

14. Cancino Chacón, C., Lattner, S., and Grachten, M. (2014a). Developing Tonal Perception Through Unsupervised Learning. In *Proceedings of the 15th International Society for Music Information Retrieval Conference (ISMIR 2014)*, pages 195–200

15. Grachten, M., Cancino Chacón, C. E., and Widmer, G. (2014). Analysis and Prediction of Expressive Dynamics Using Bayesian Linear Models. In *Proceedings of the 1st International Workshop on Computer and Robotic Systems for Automatic Music Performance (SAMP 14)*, pages 545–552, Venice, Italy

16. Tschiatschek, S., Cancino Chacón, C. E., and Pernkopf, F. (2013). Bounds for Bayesian network classifiers with reduced precision parameters. In *Proceedings of the 2013 International Conference on Acoustics, Speech and Signal Processing (ICASSP 2013)*, pages 3357–3361, Vancouver, Canada. IEEE

EXTENDED ABSTRACTS

1. Cancino-Chacón, C. and Grachten, M. (2018). A Computational Study of the Role of Tonal Tension in Expressive Piano Performance. In *Proceedings of the 15th International Conference on Music Perception and Cognition (ICMPC15 ESCOM10)*, Graz, Austria

2. Bishop, L., Cancino-Chacón, C. E., and Goebl, W. (2018). Visual Signals between Improvisers Indicate Attention rather than Intentions. In *Proceedings of the 15th International Conference on Music Perception and Cognition (ICMPC15 ESCOM10)*, Graz, Austria

3. Cancino-Chacón, C., Bonev, M., Durand, A., Grachten, M., Arzt, A., Bishop, L., Goebl, W., and Widmer, G. (2017a). The ACCompanion v0.1: An Expressive Accompaniment System. In *Late Breaking/ Demo, 18th International Society for Music Information Retrieval Conference (ISMIR 2017)*, Suzhou, China

4. Bishop, L., Cancino-Chacón, C., and Goebel, W. (2017). Mapping Visual Attention of Duo Musicians During Rehearsal of Temporally-Ambiguous Music. In *Proceedings of the International Symposium on Performance Science (ISPS 2017)*, Reykjavik, Iceland

5. Cancino Chacón, C. E. and Grachten, M. (2016). The Basis Mixer: A Computational Romantic Pianist. In *Late Breaking/ Demo, 17th International Society for Music Information Retrieval Conference (ISMIR 2016)*, New York, NY, USA

TECHNICAL REPORTS

1. Cancino-Chacón, C. E. and Grachten, M. (2016). Rendering Expressive Performances of Musical Pieces Through Sampling From Generative Probabilistic Models. Technical Report OFAI-TR-2014-01, Austrian Research Institute for Artificial Intelligence, Vienna, Austria

2. Grachten, M. and Cancino Chacón, C. E. (2015). Strategies for Conceptual Change in Convolutional Neural Networks. Technical Report OFAI-TR-2015-04, Austrian Research Institute for Artificial Intelligence, Vienna, Austria

3. Cancino Chacón, C. E., Grachten, M., and Widmer, G. (2014b). Bayesian Linear Basis Models with Gaussian Priors for Musical Expression. Technical Report OFAI-TR-2014-12, Austrian Research Institute for Artificial Intelligence, Vienna, Austria